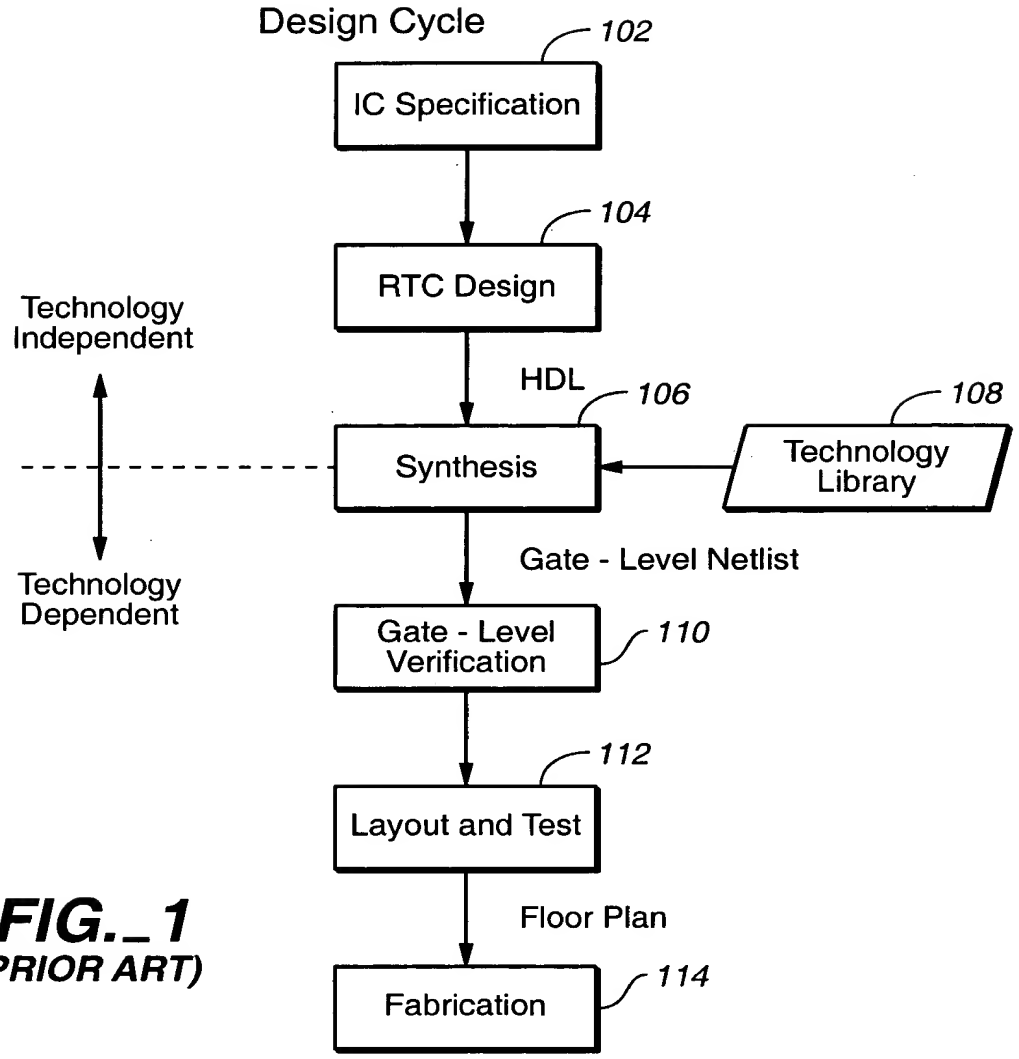


FIG._1
(PRIOR ART)

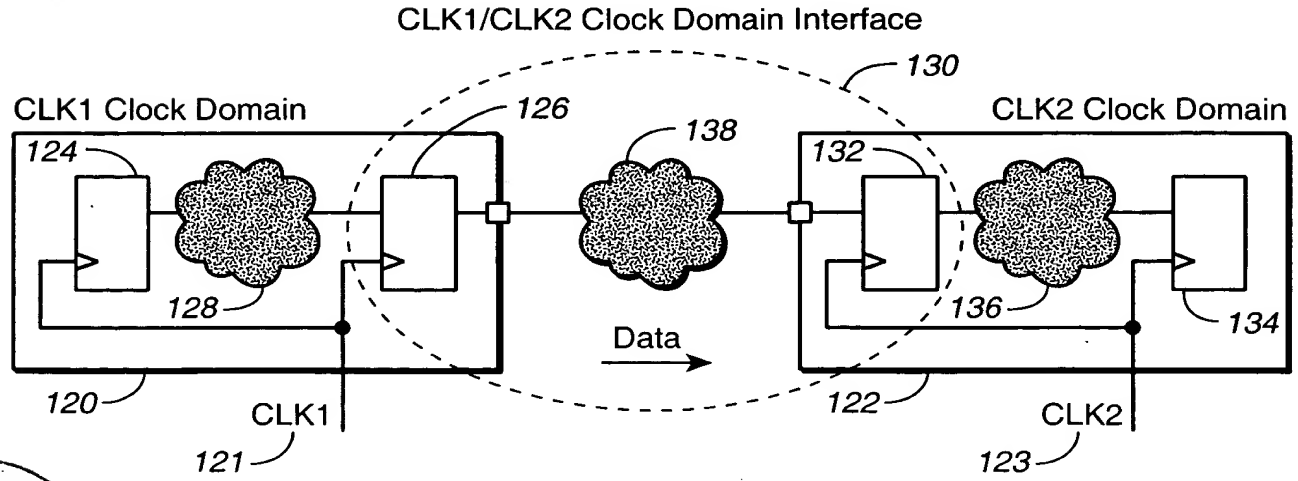
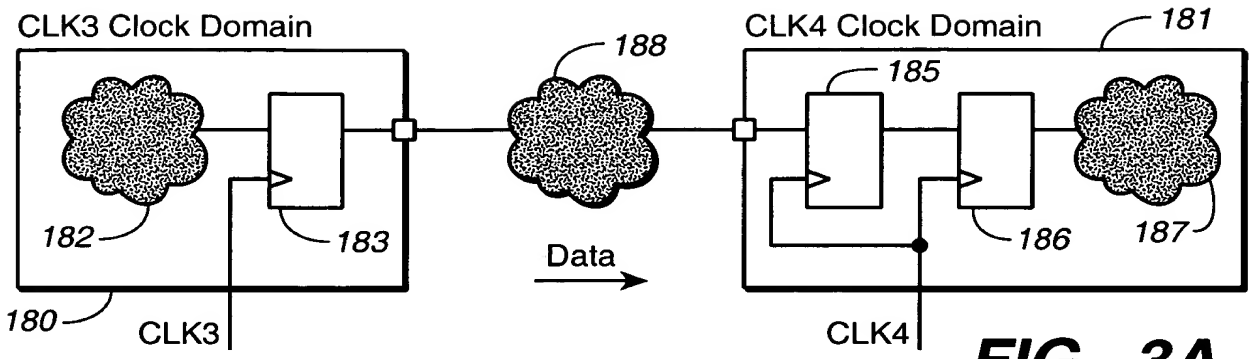
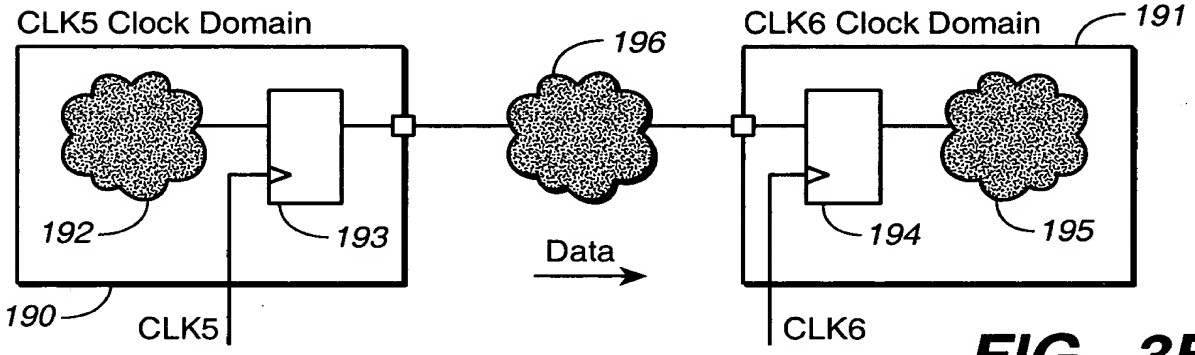


FIG._2



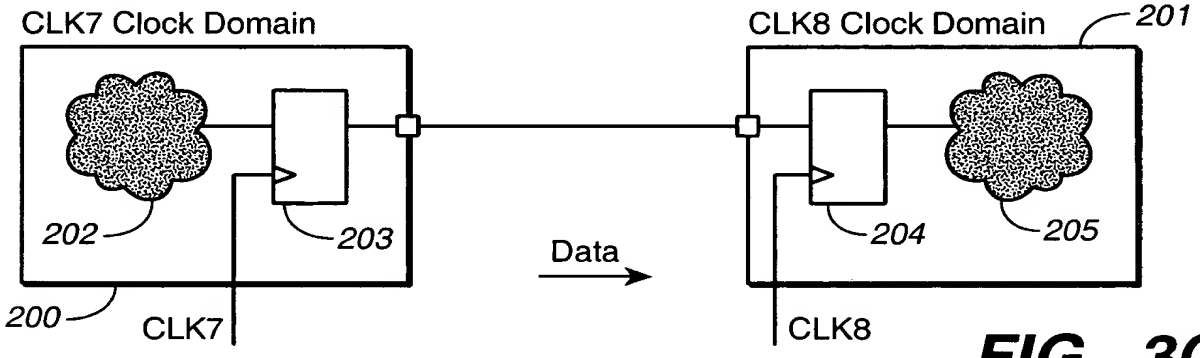
: Asynchronous clock domains

FIG._3A



: Related clock domains with coincident clock edges

FIG._3B



: Related clock domains with sequenced clock edges

FIG._3C

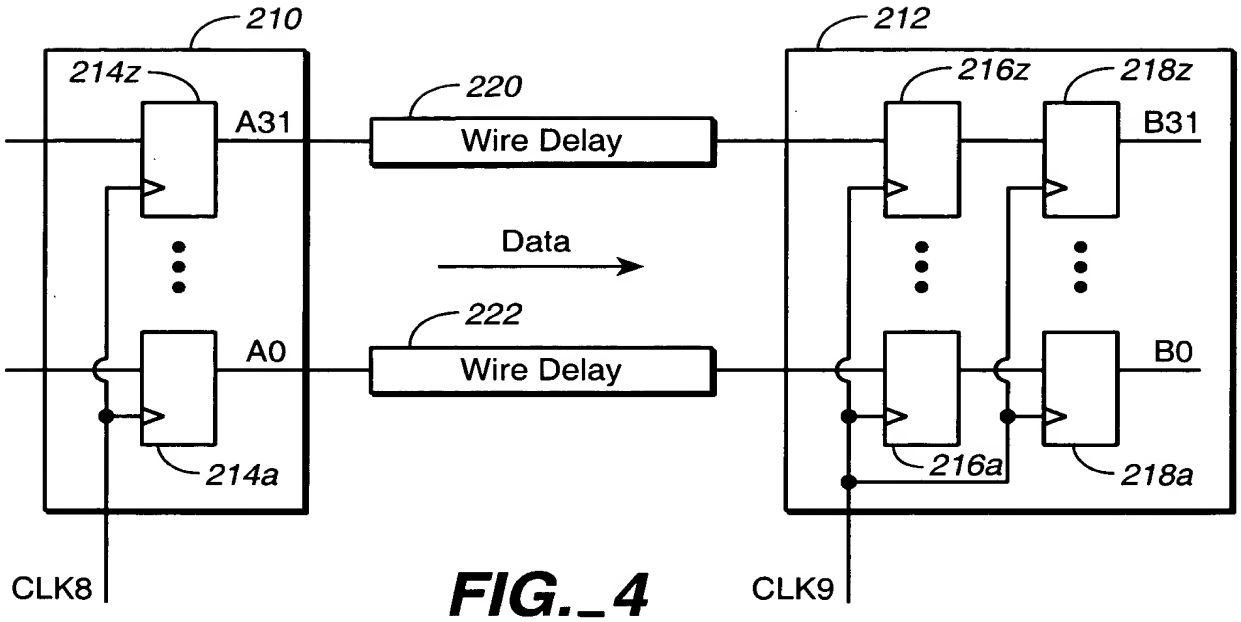


FIG._4

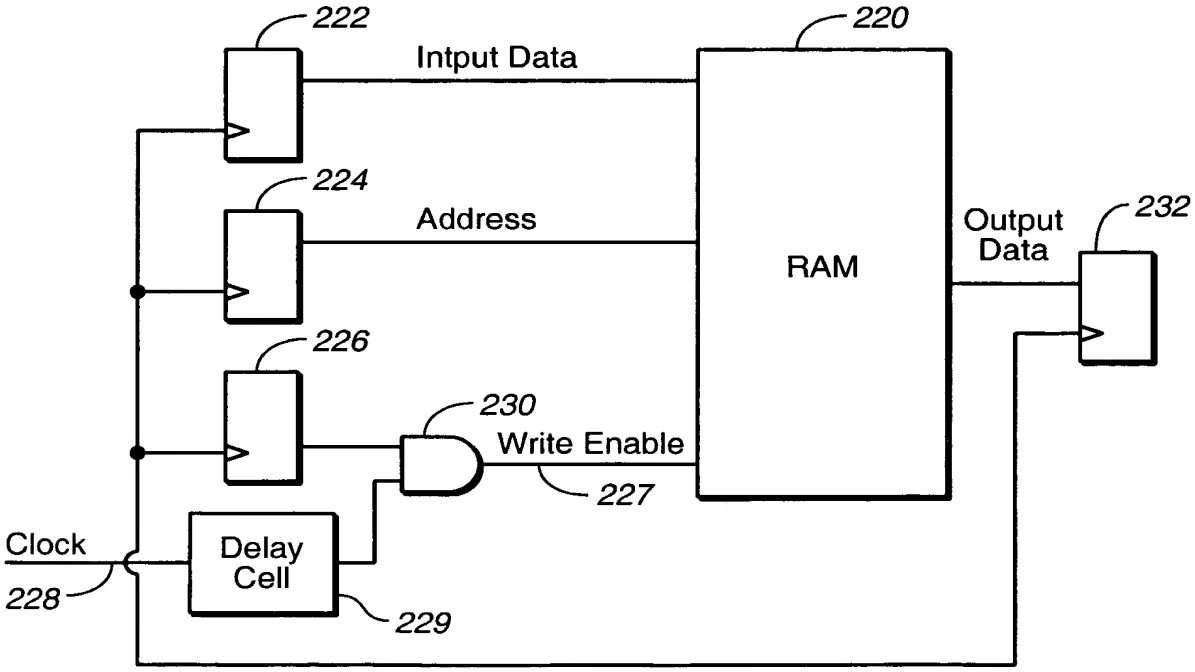
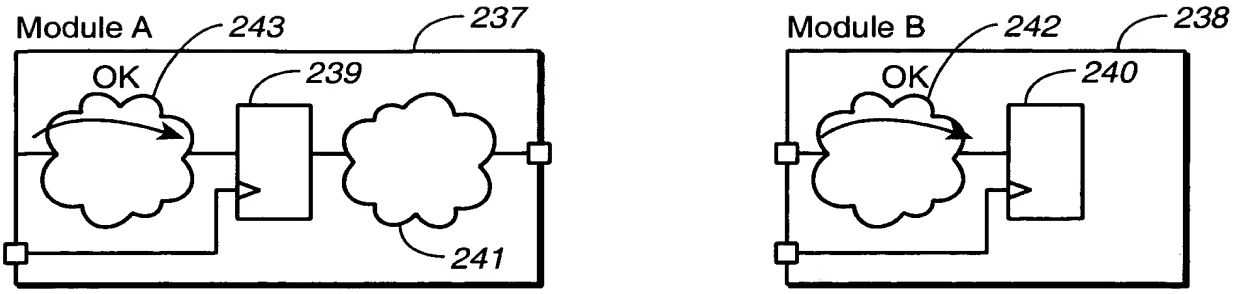
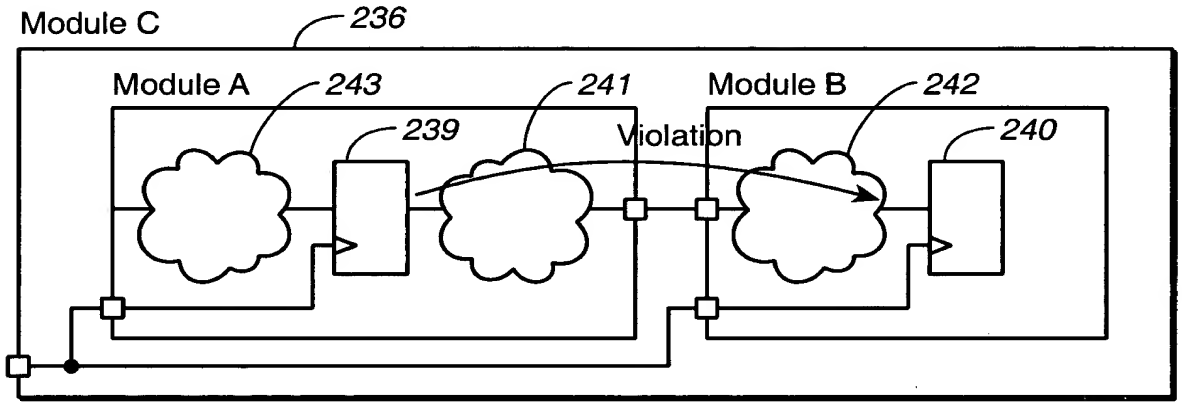


FIG._5

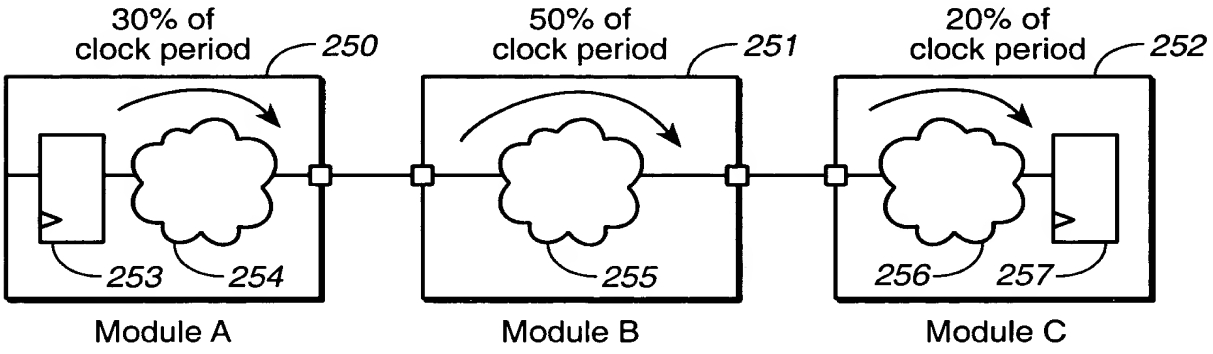


1. Exploratory sysnthesis on individual modules meets clock constrains.



2. Violating paths show up when modules are put together

FIG._6



Modules A,B and C compiled independently

FIG._7

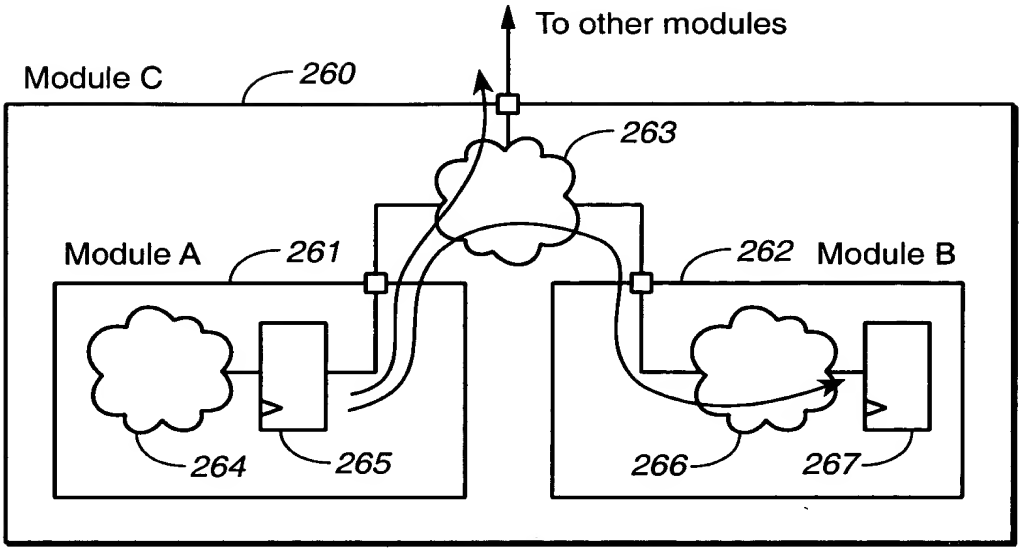
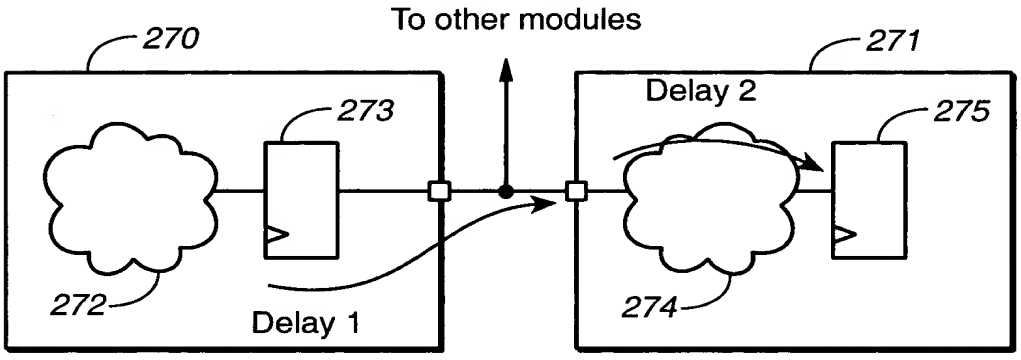
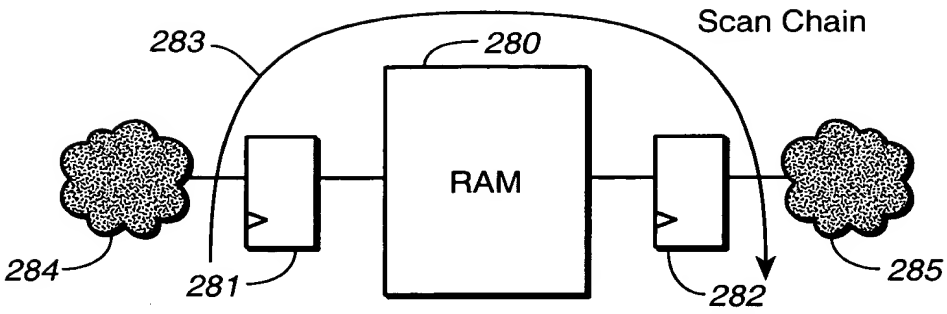


FIG._8



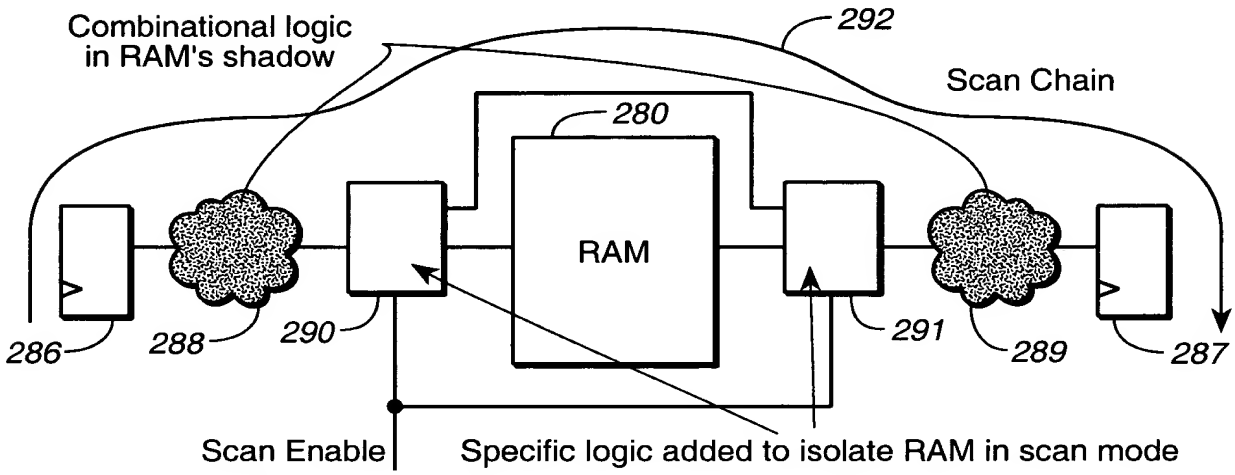
Delay 1<<Delay 2

FIG._9



With no combinational logic in RAM's shadow

FIG._10A



With combinational logic in RAM's shadow

FIG._10B

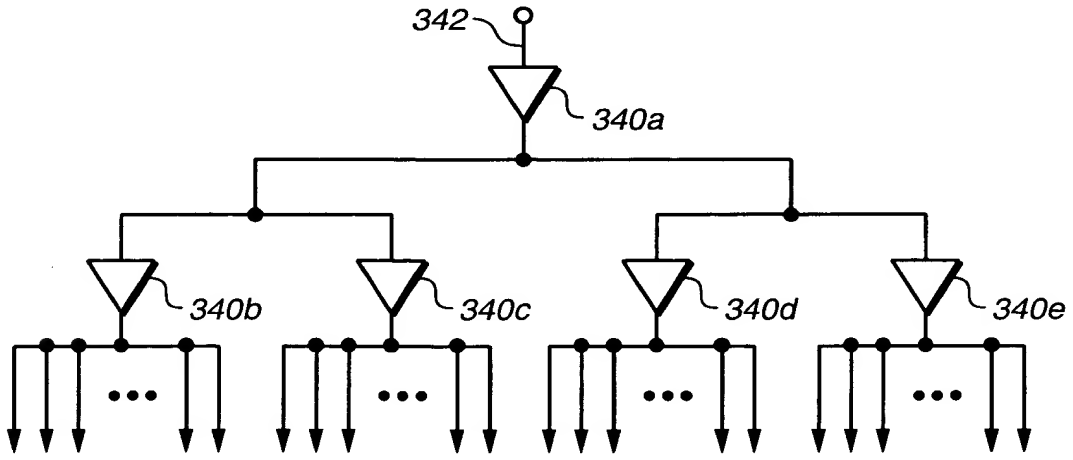


FIG._11

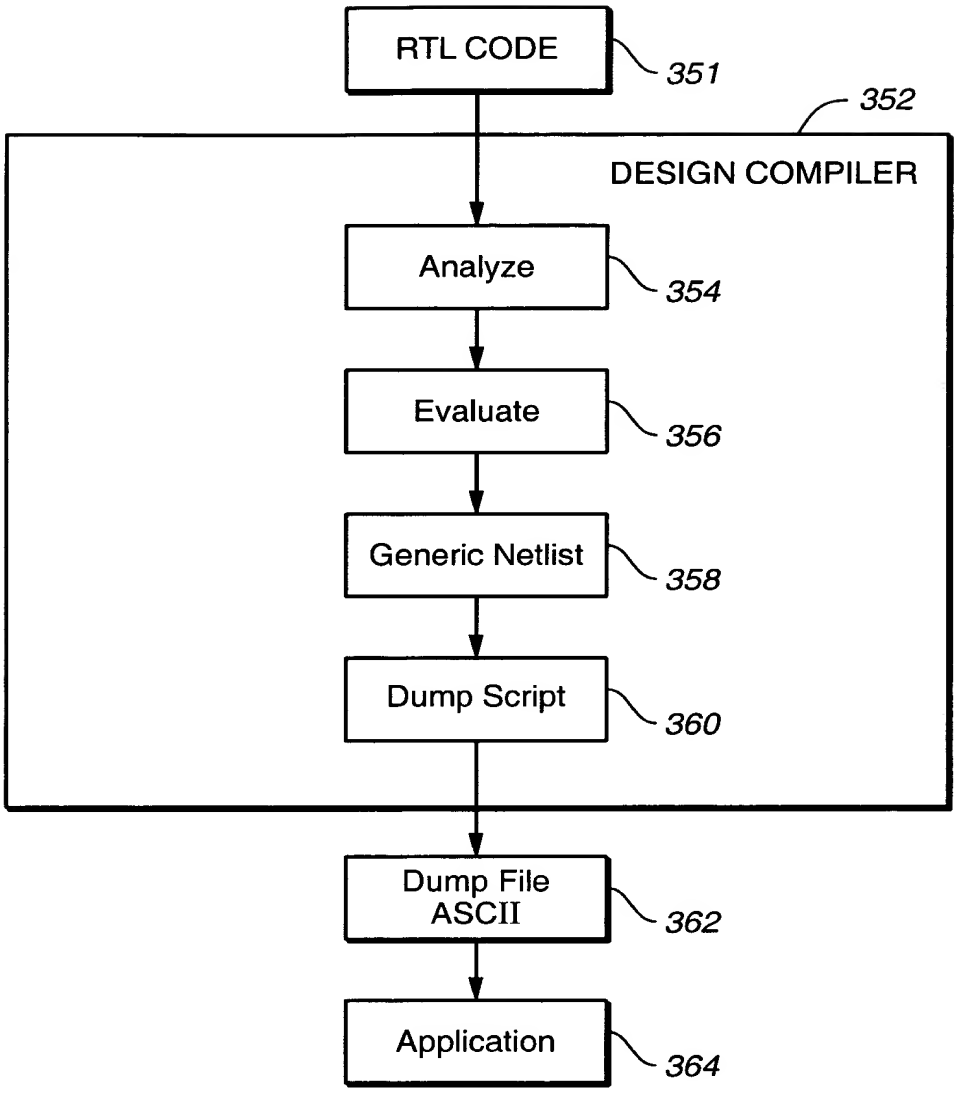


FIG._12

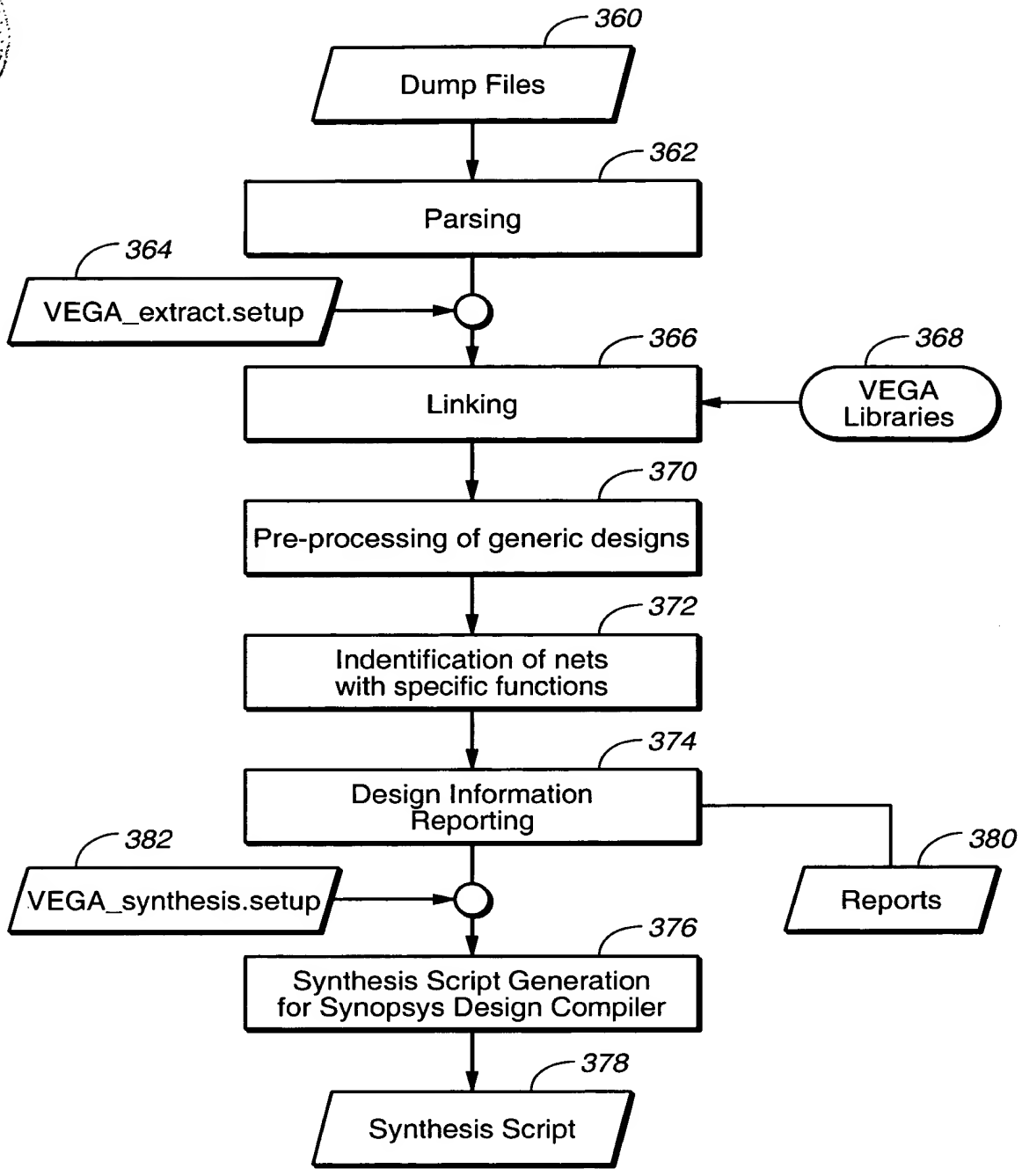


FIG._13

VEGA FLOW

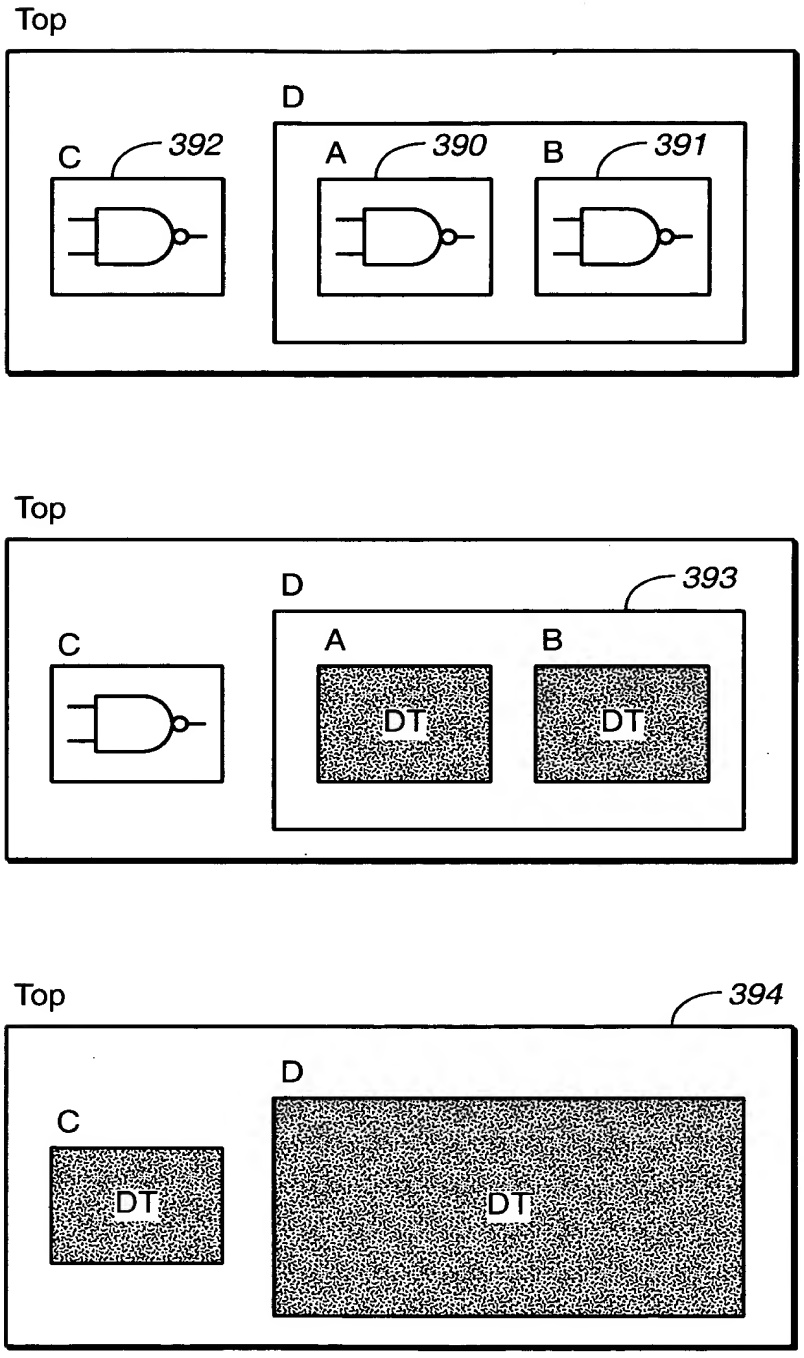


FIG._14 : Bottom-up synthesis.

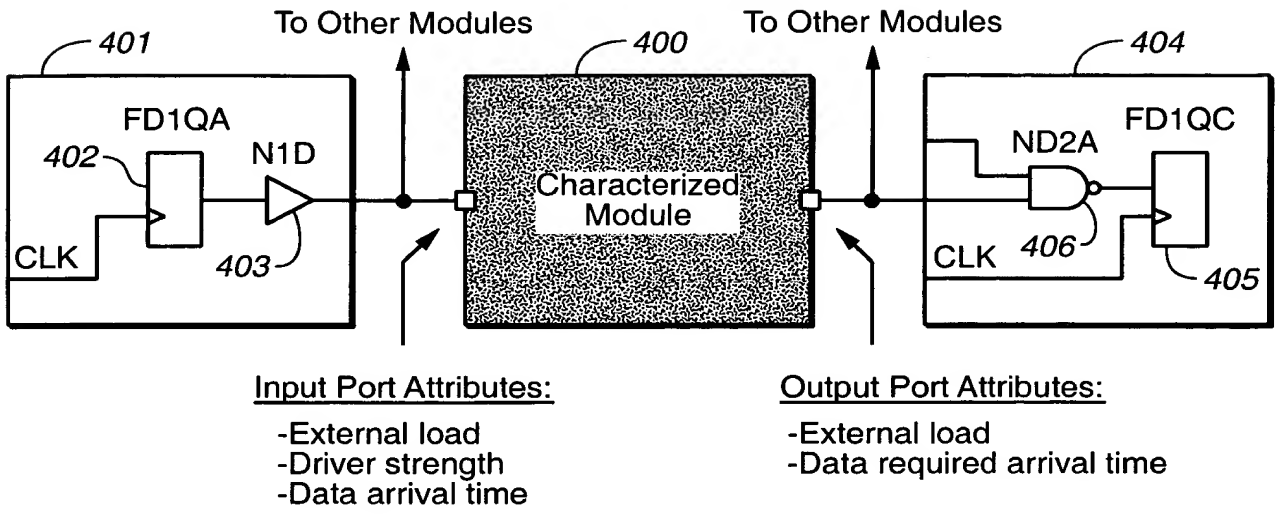


FIG._15 :Characterization.

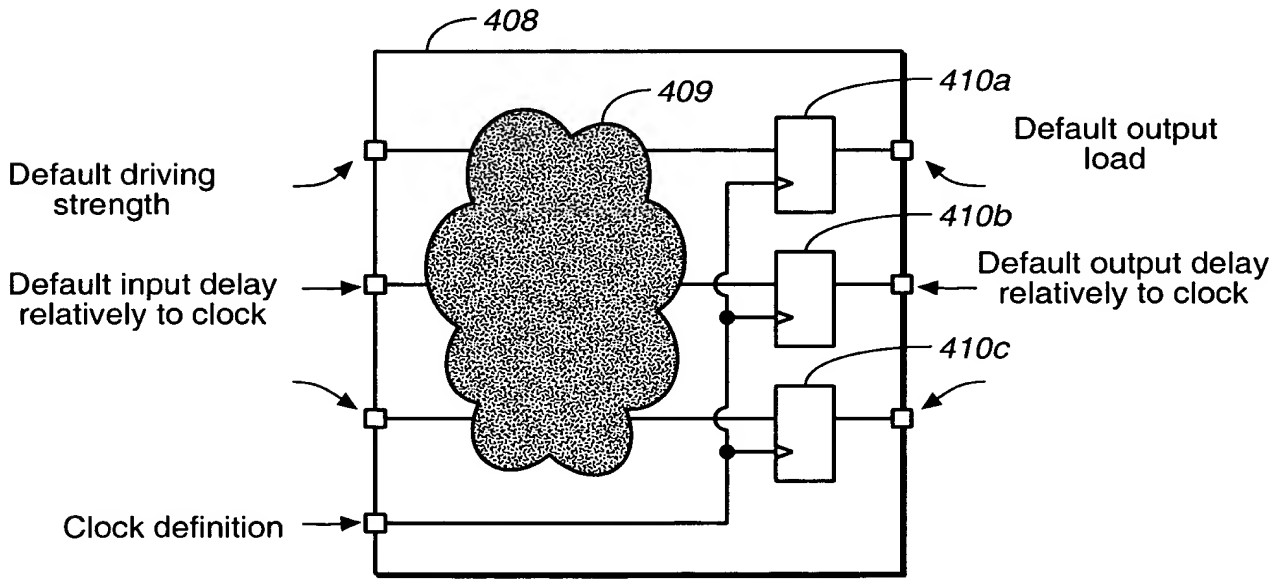
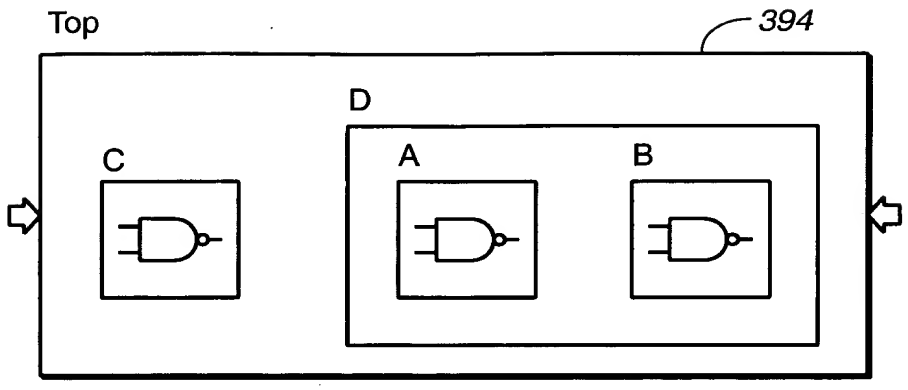
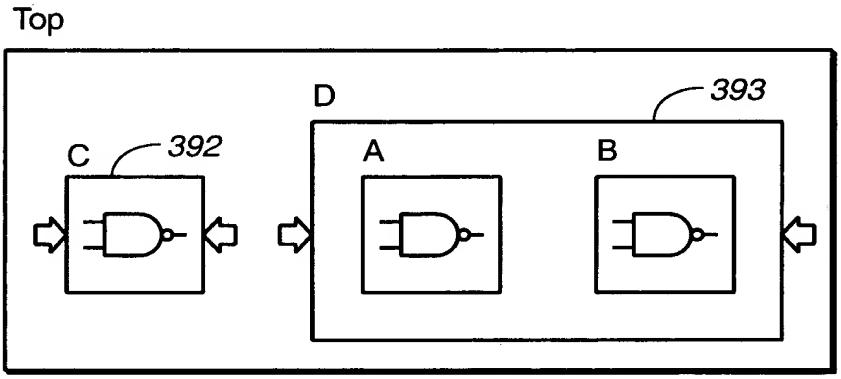


FIG._17 :Default constraints used for initial mapping.



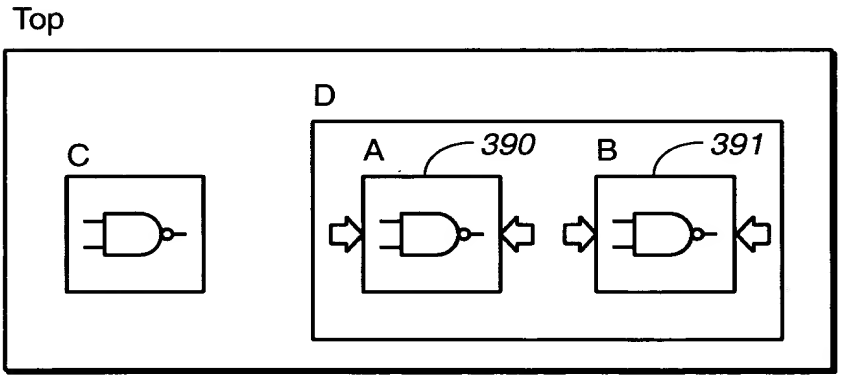
Step #1

: Constraints are set on top-level module (operating conditions, clock definitions, ect.).



Step #2

: Constraints are derived on modules C and D.



Step #3

: Constraints are derived on leaf modules A and B.

FIG._16 : Top-down characterization.

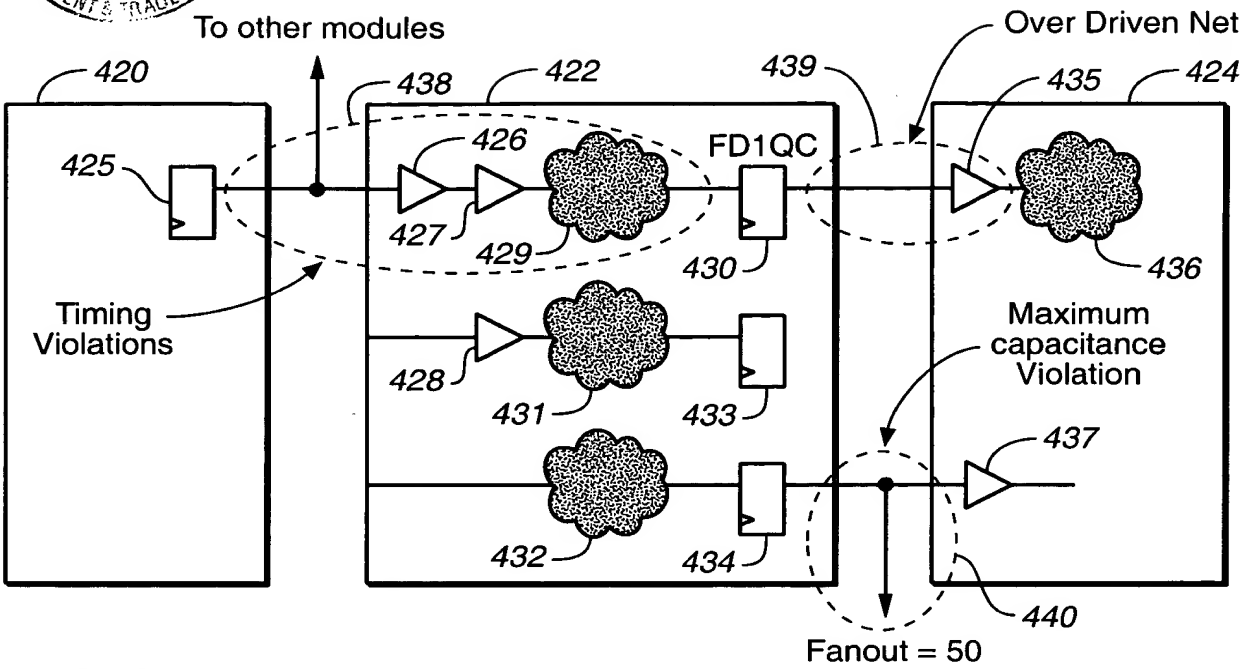


FIG._18: Results after initial mapping.

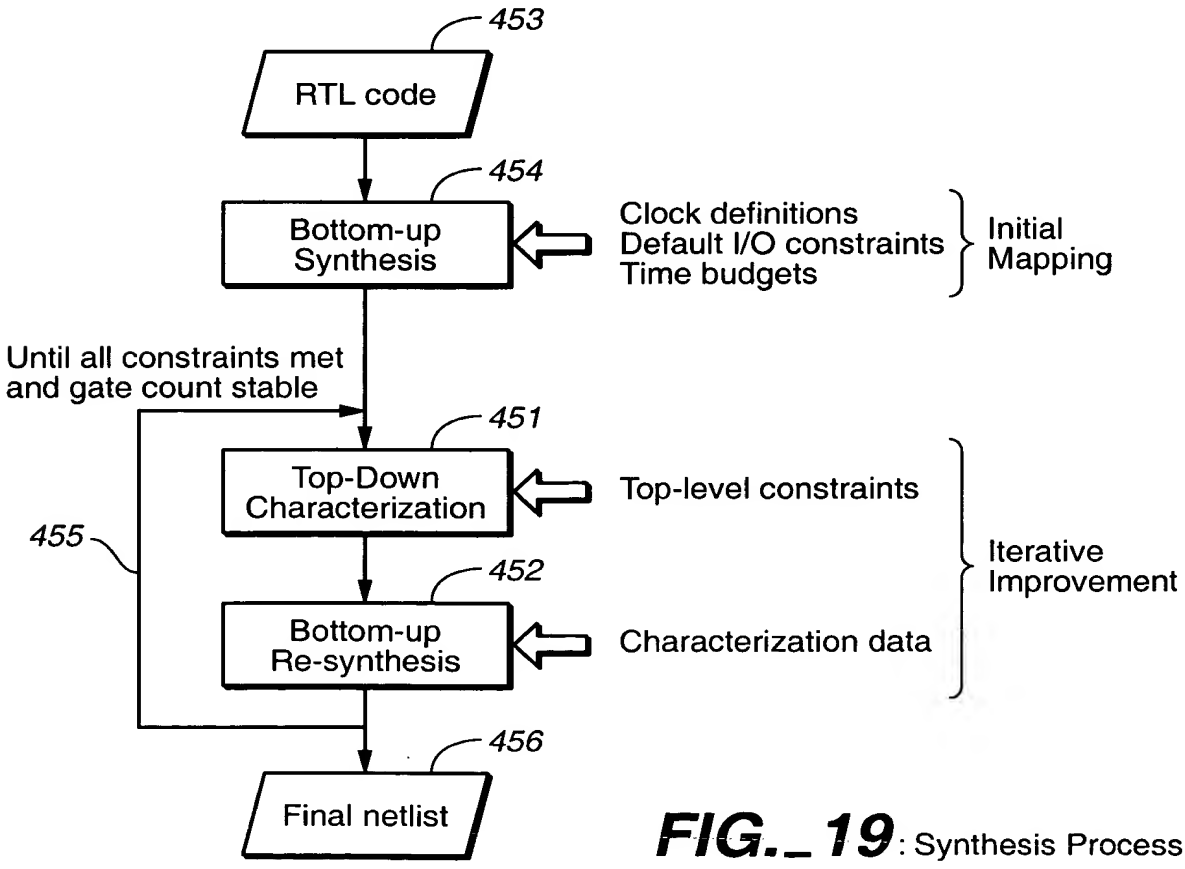
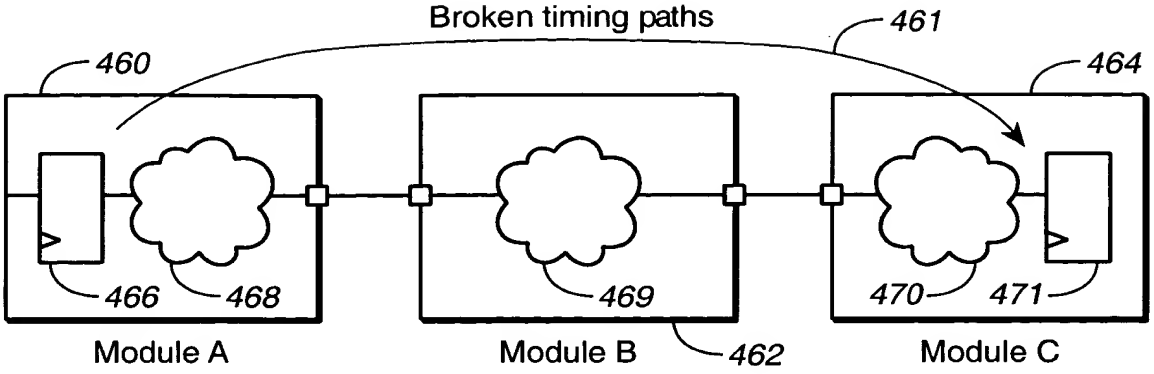
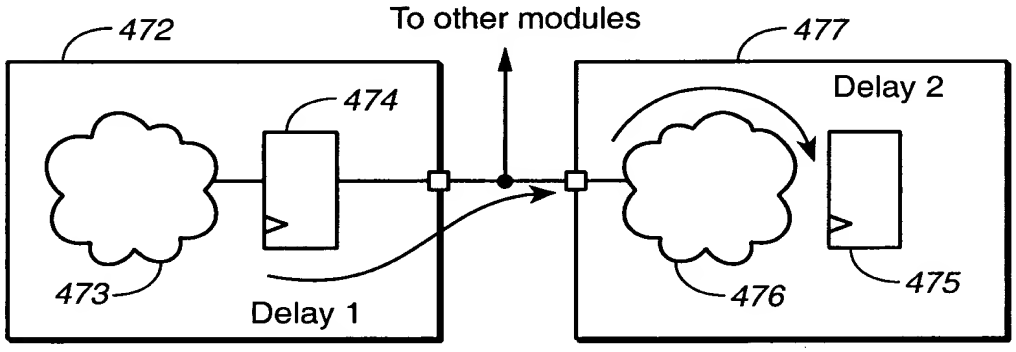


FIG._19: Synthesis Process



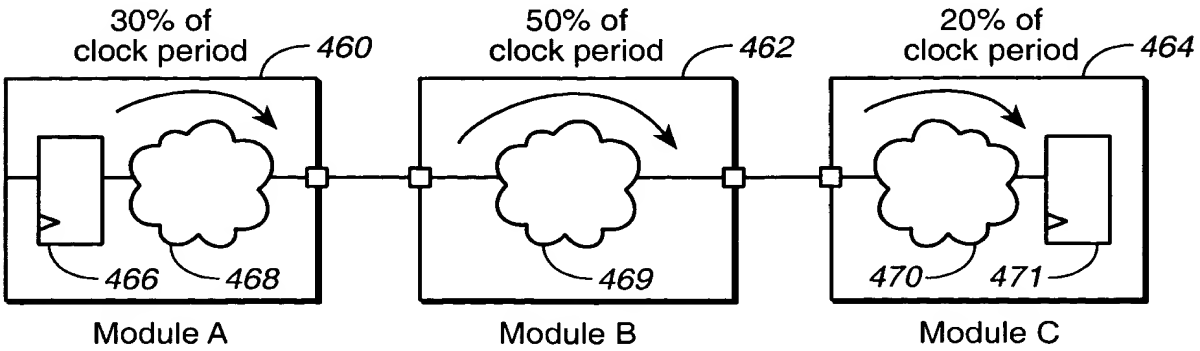
Modules A, B and C synthesized independently from each other

FIG._20A : Broken timing paths



Delay 1<<Delay 2

FIG._20B : In the absence of broken timing paths



Modules A, B and C synthesized independently from each other

FIG._20C : Time budgets.

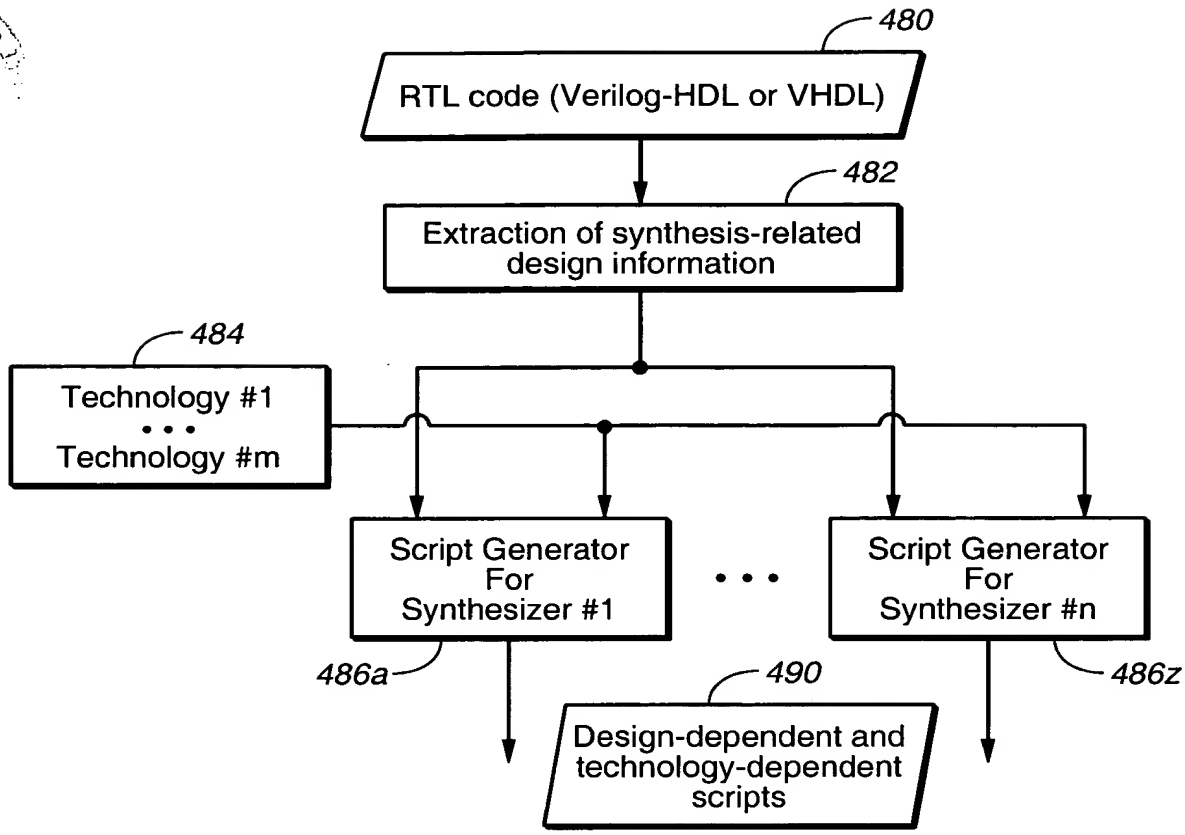


FIG._21 : Automatic Script Generation

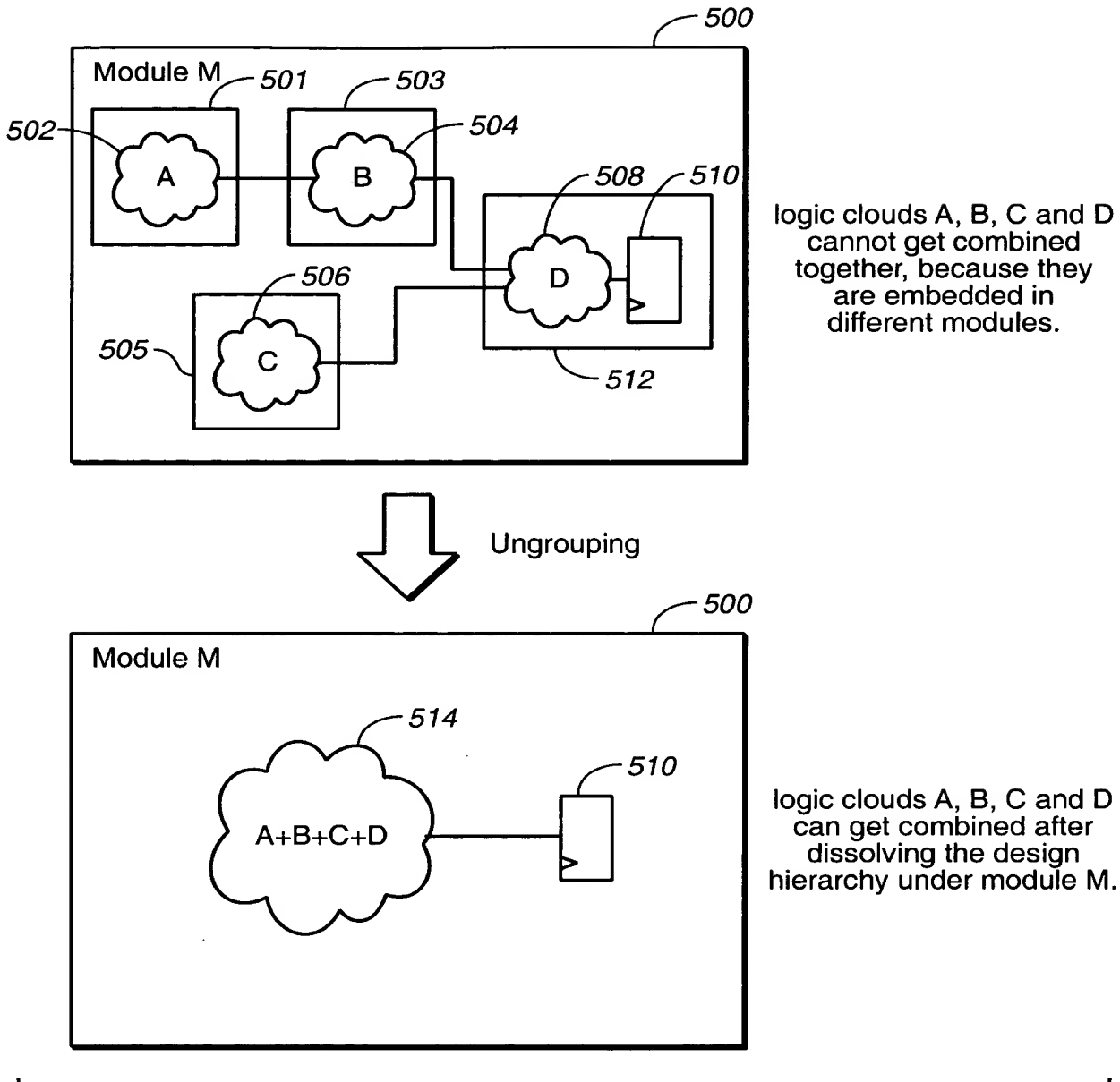


FIG._22 : Hierarchy ungrouping.

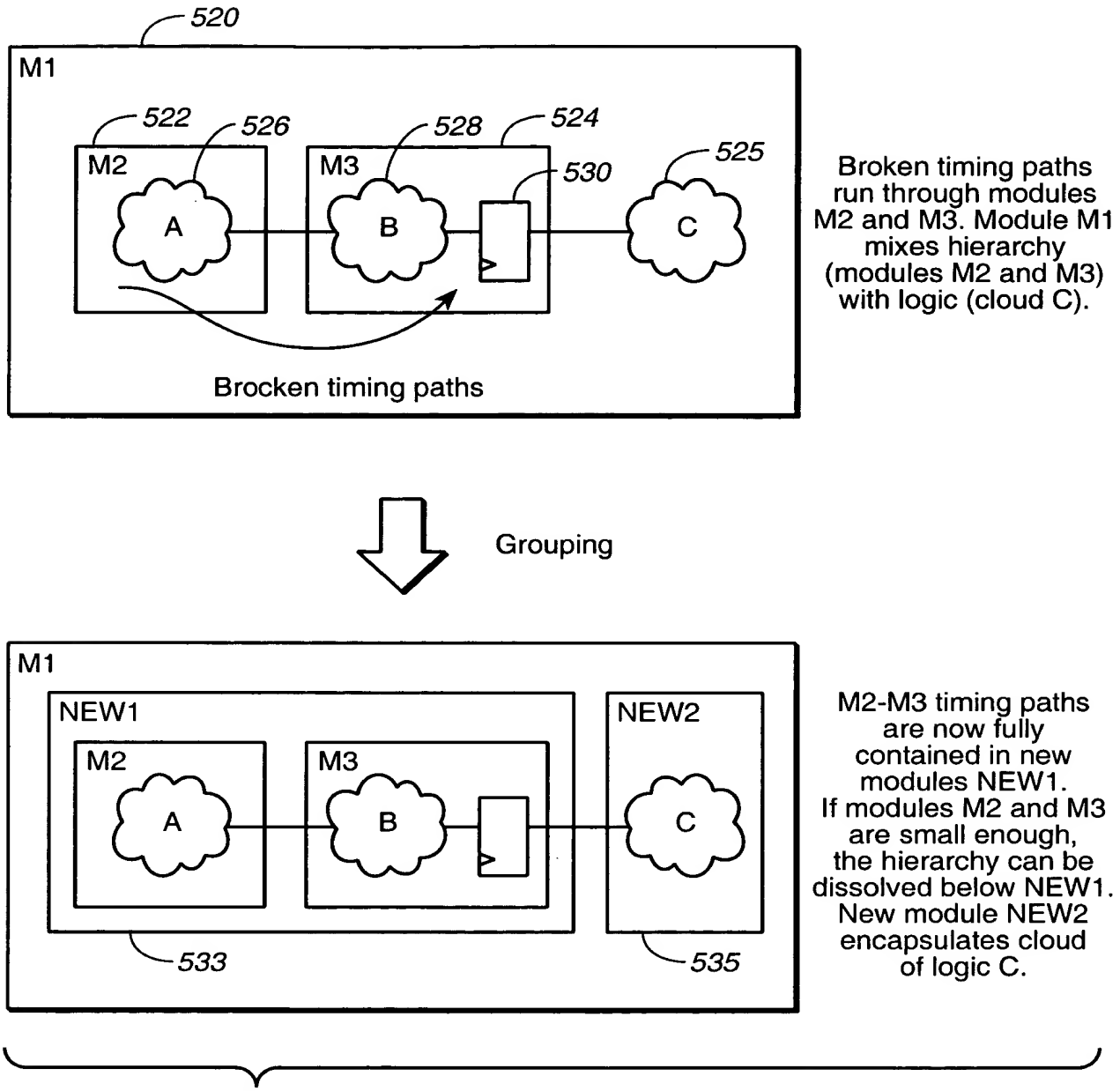


FIG._23 : Hierarchy grouping.

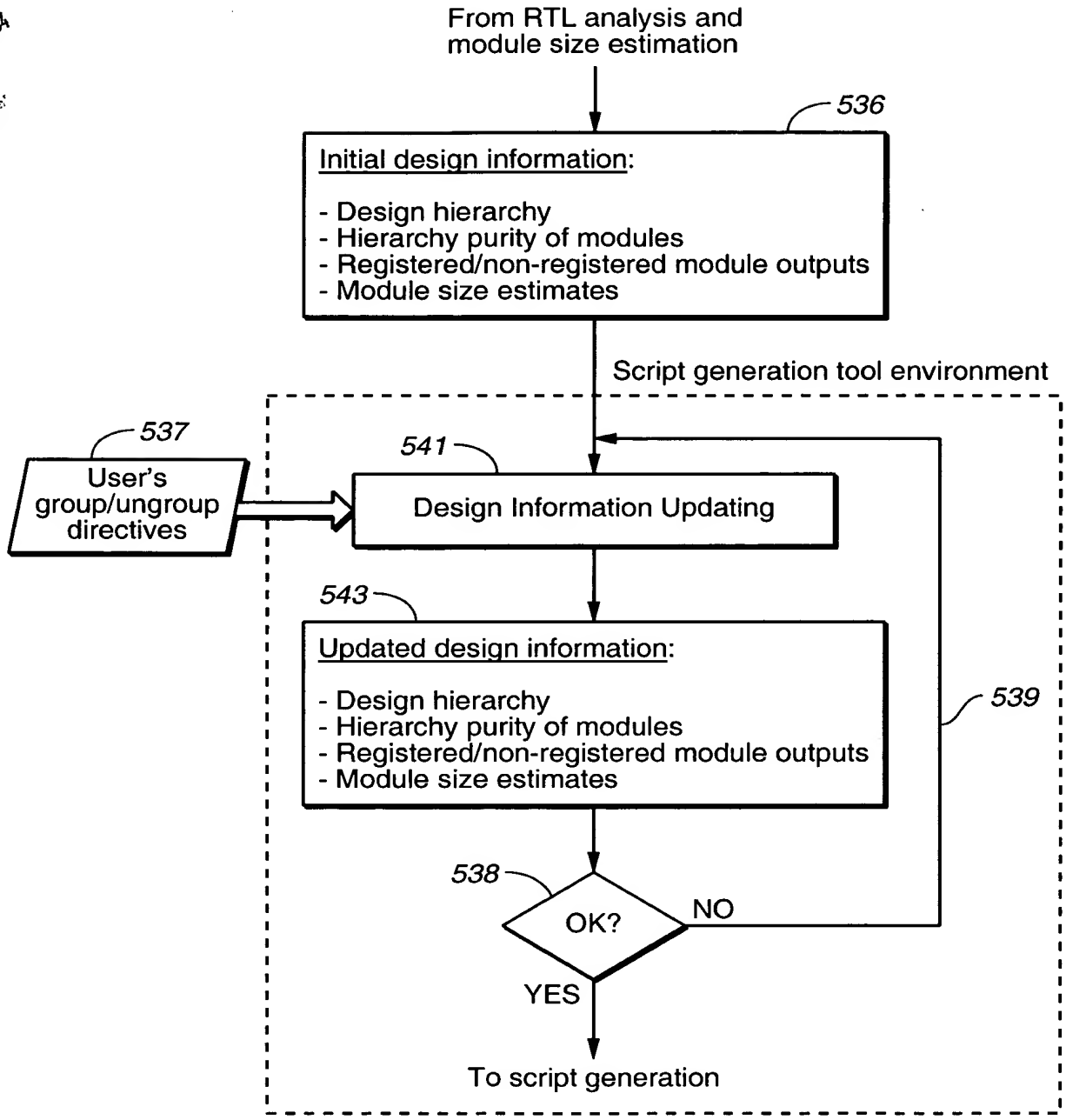


FIG._24 : Support for design hierarchy re-arrangement

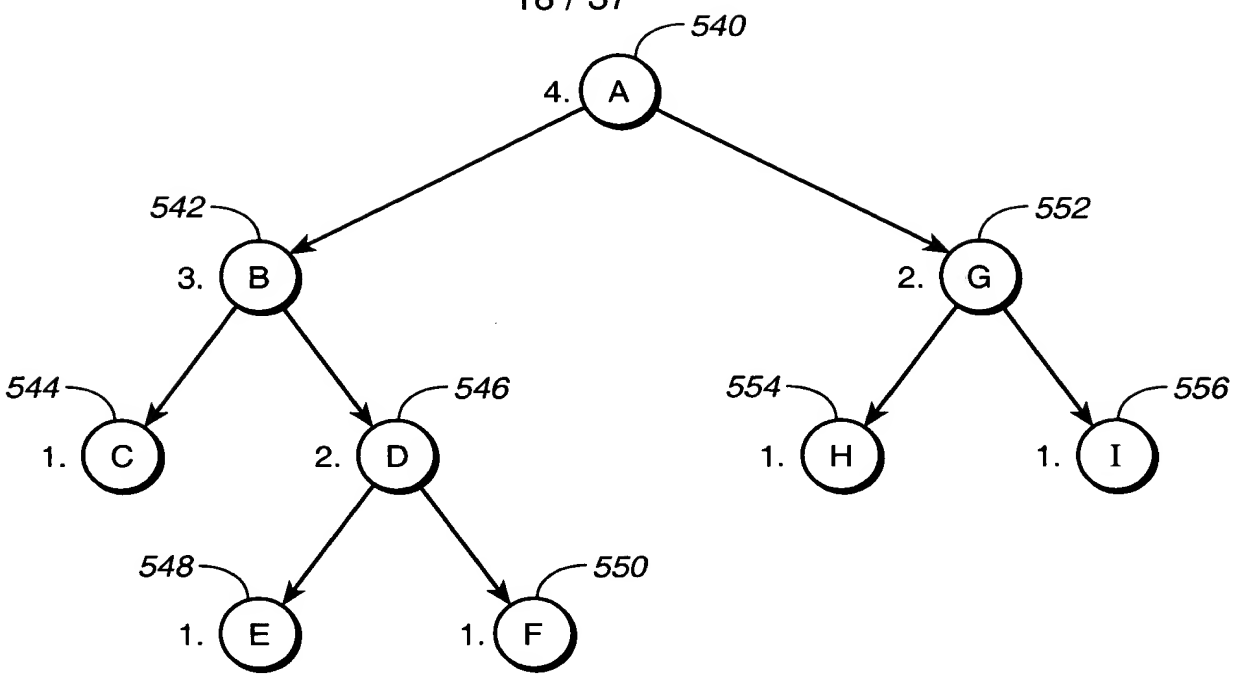


FIG._25 : Module processing order for parallel bottom-up synthesis.

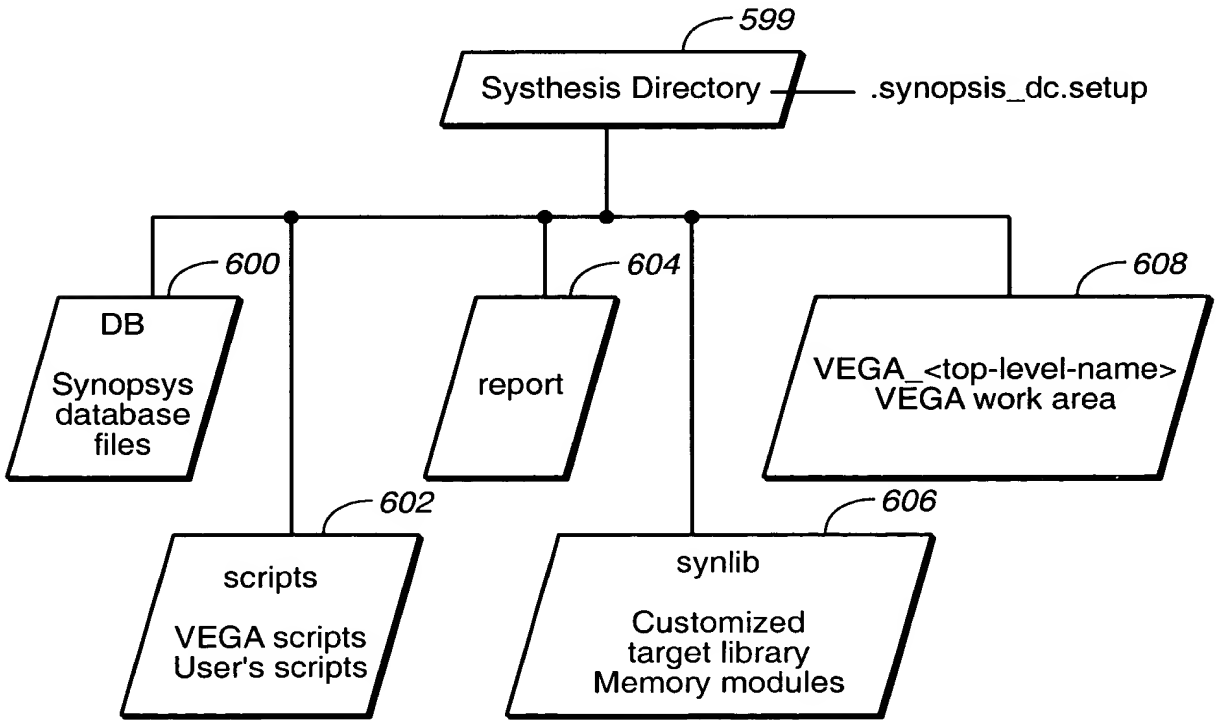


FIG._26 : Database to be used to run VEGA scripts.

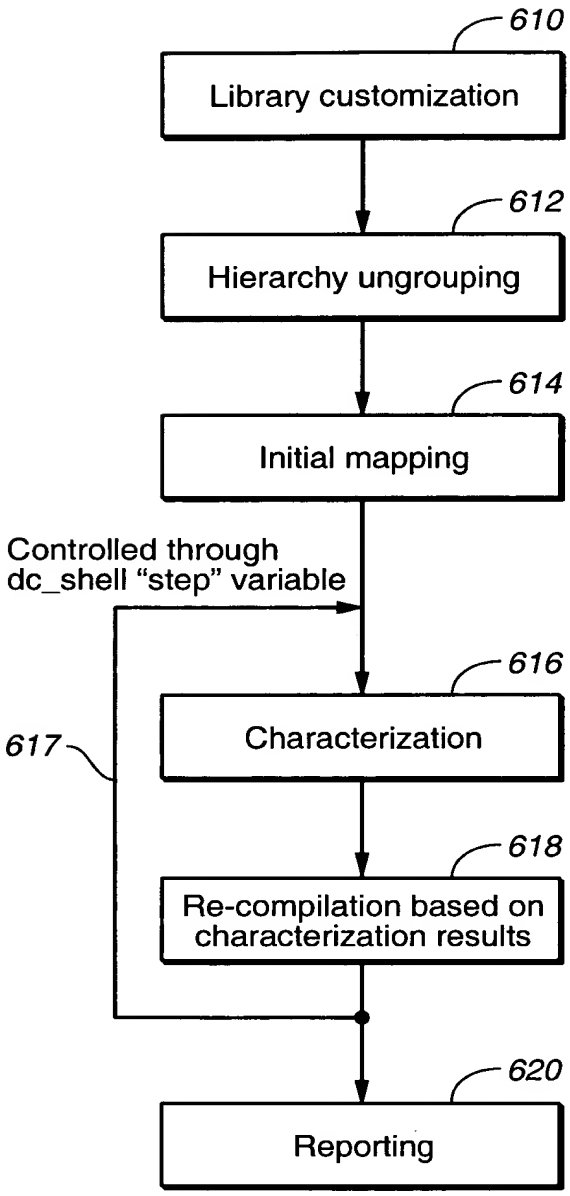


FIG._27 : Script flow implemented by VEGA

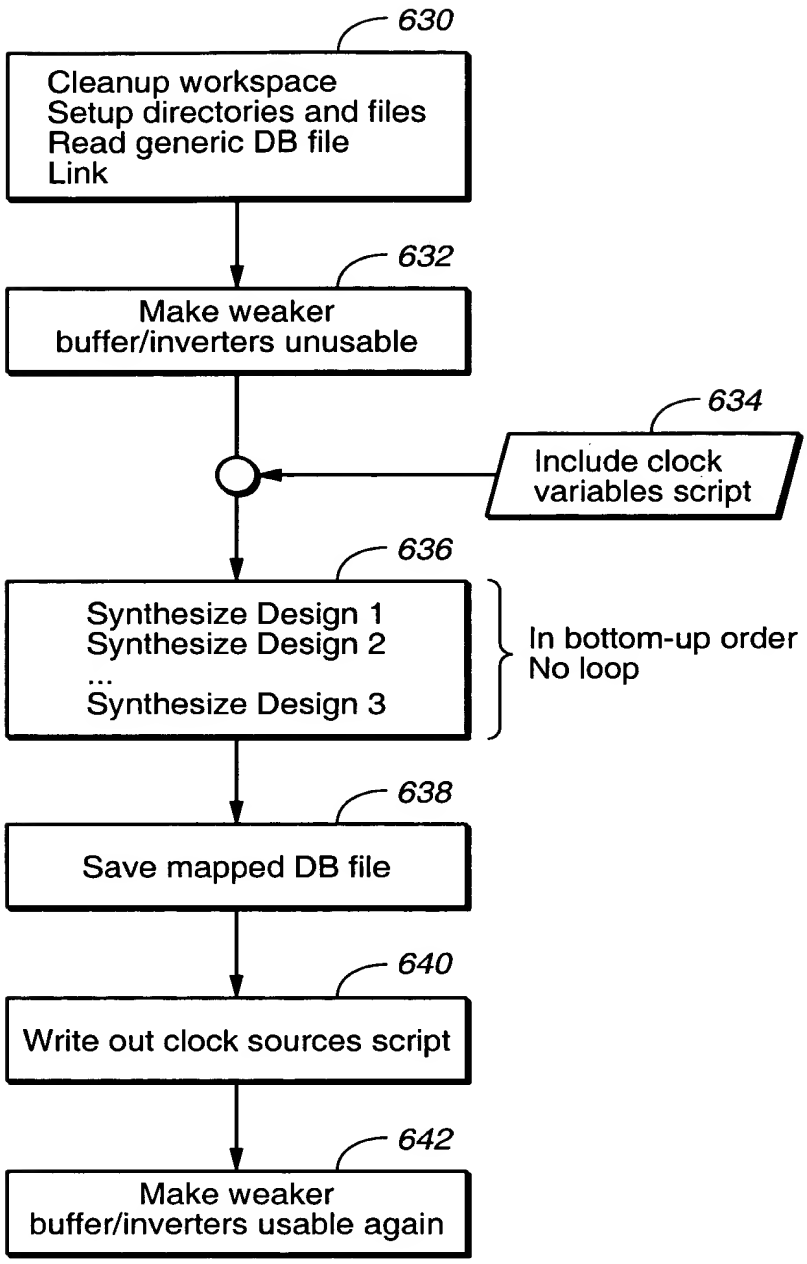


FIG._28 : Structure of initial mapping script

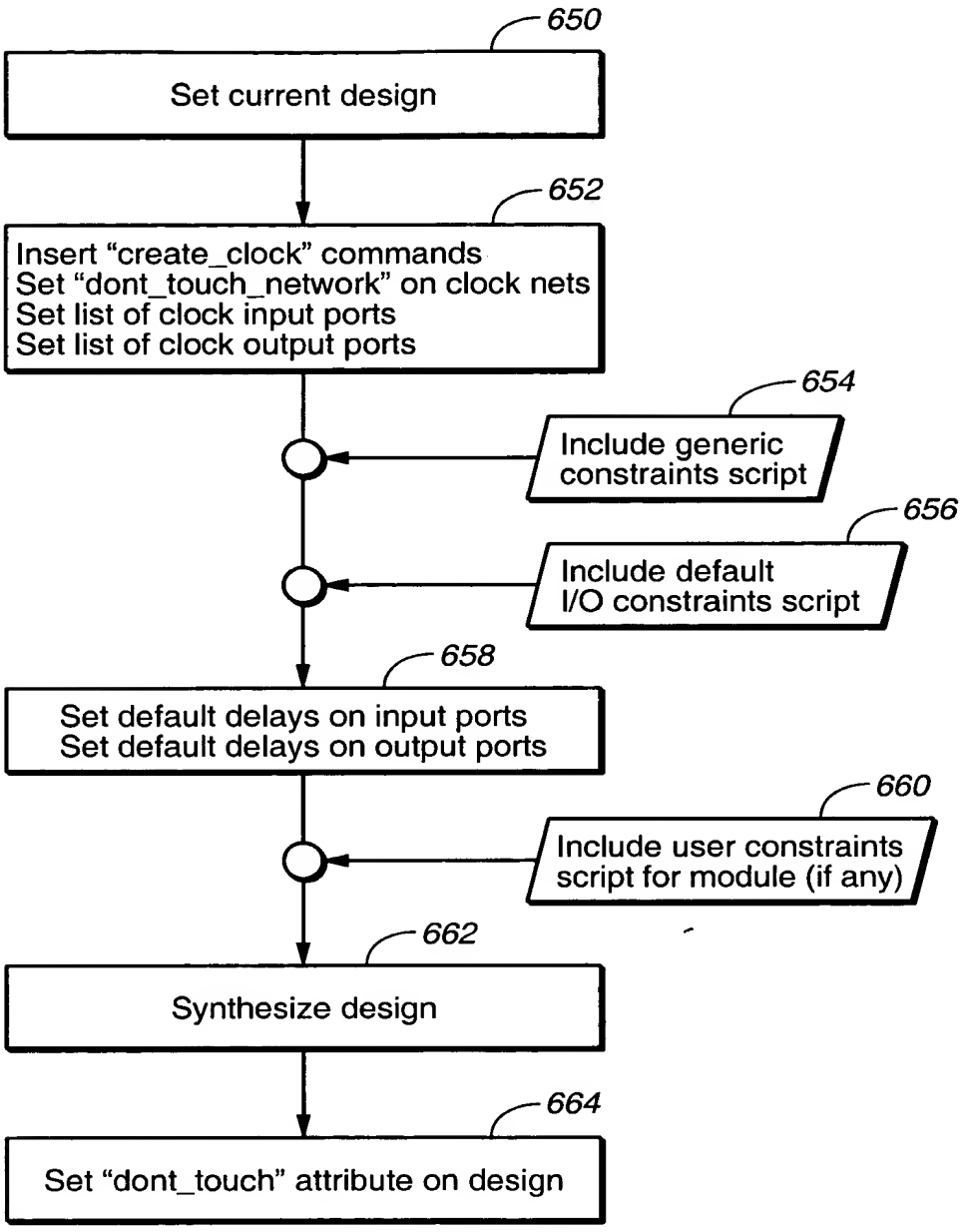


FIG._29

: Operations performed on each module by initial mapping

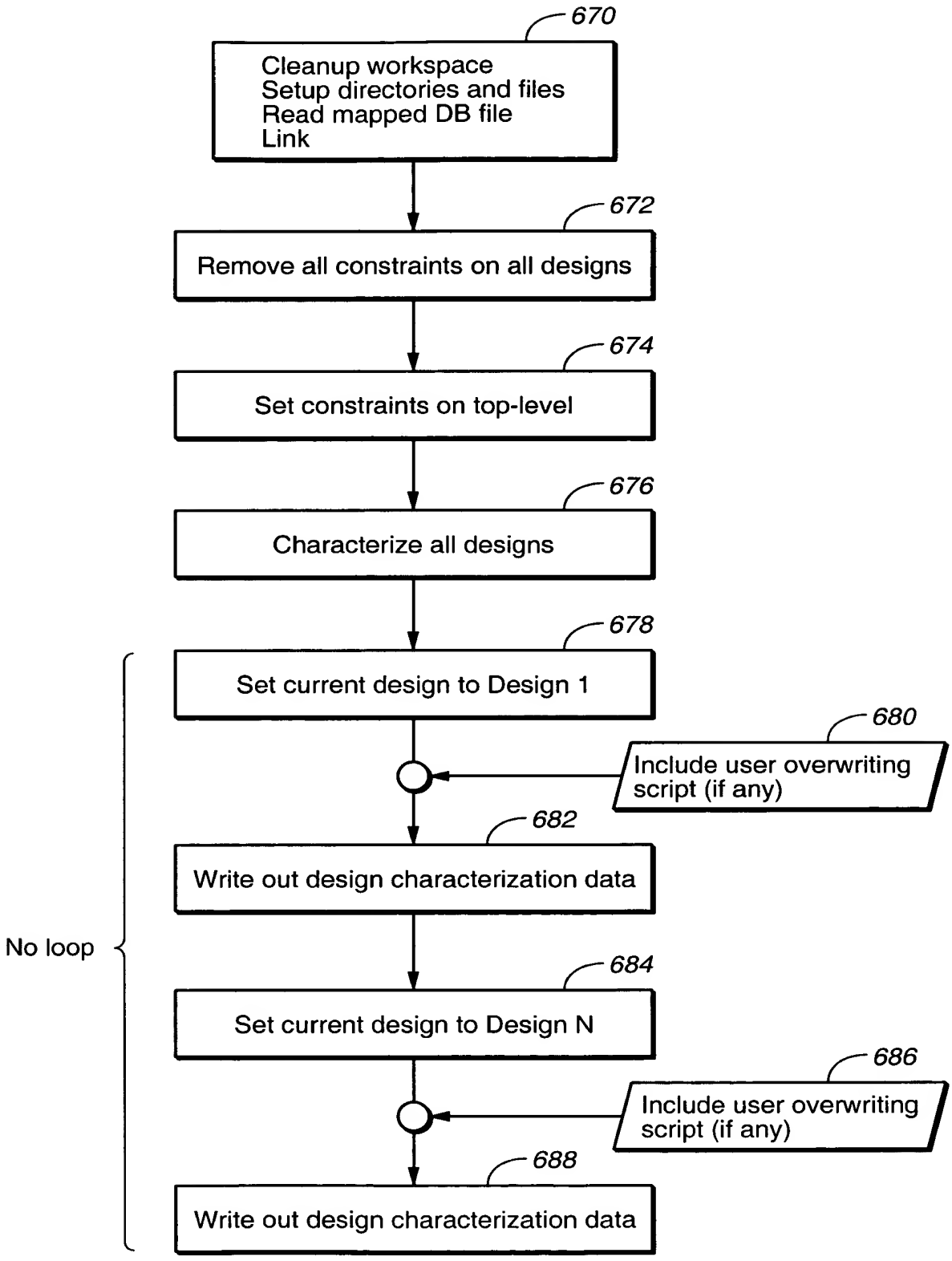


FIG._30 : Structure of characterization script

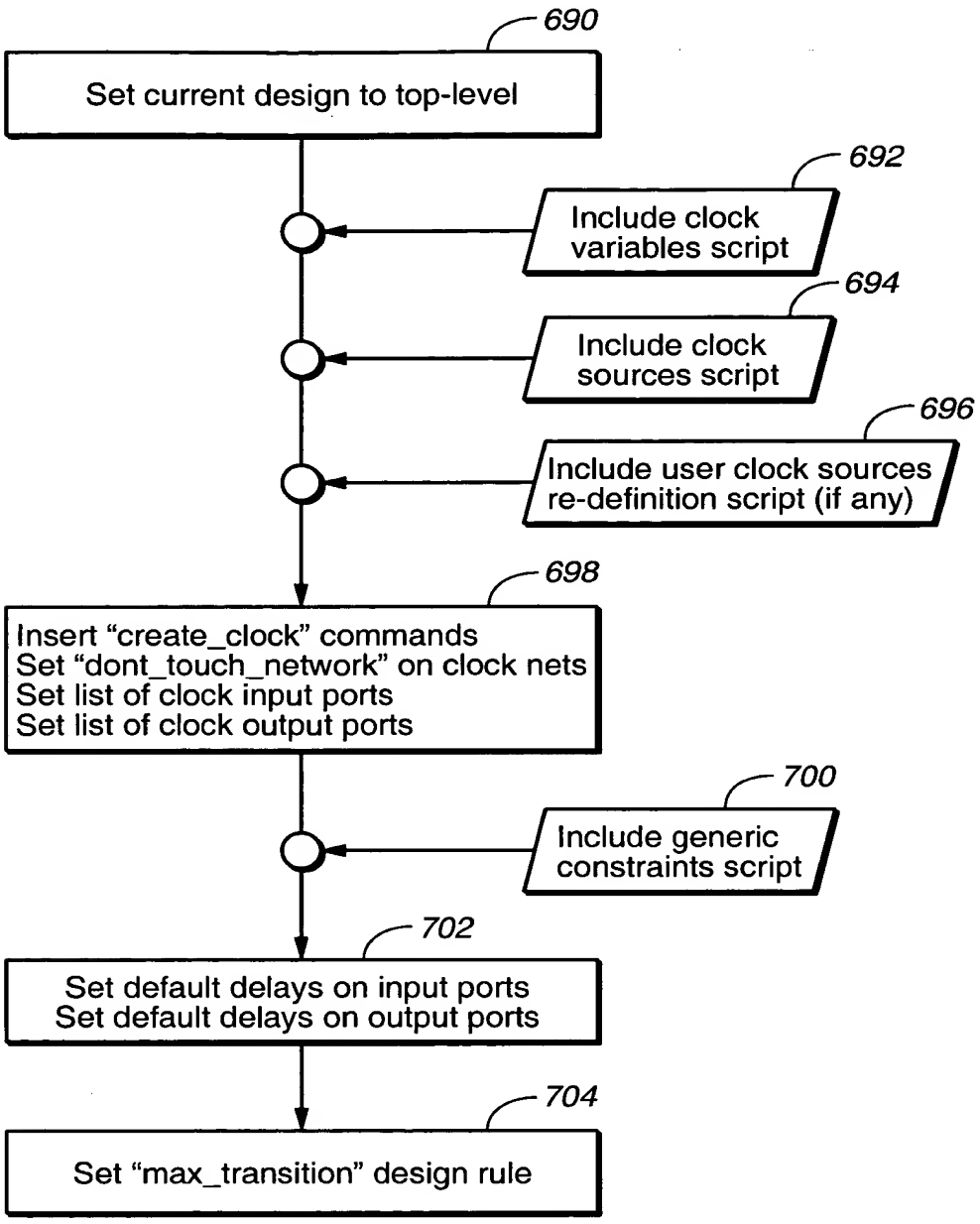


FIG._31 : Structure of constraints setting on top-level

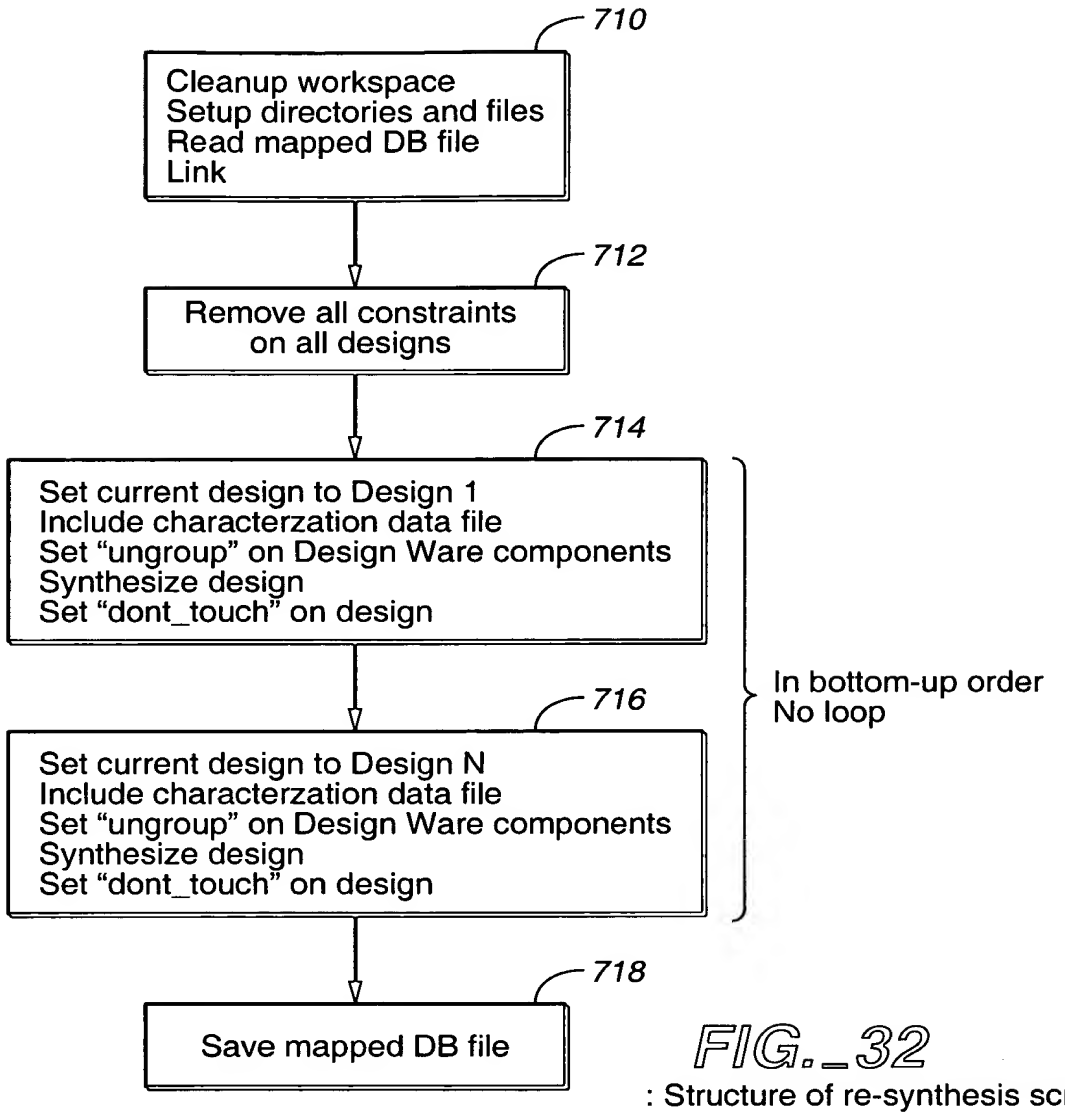


FIG._32
: Structure of re-synthesis script

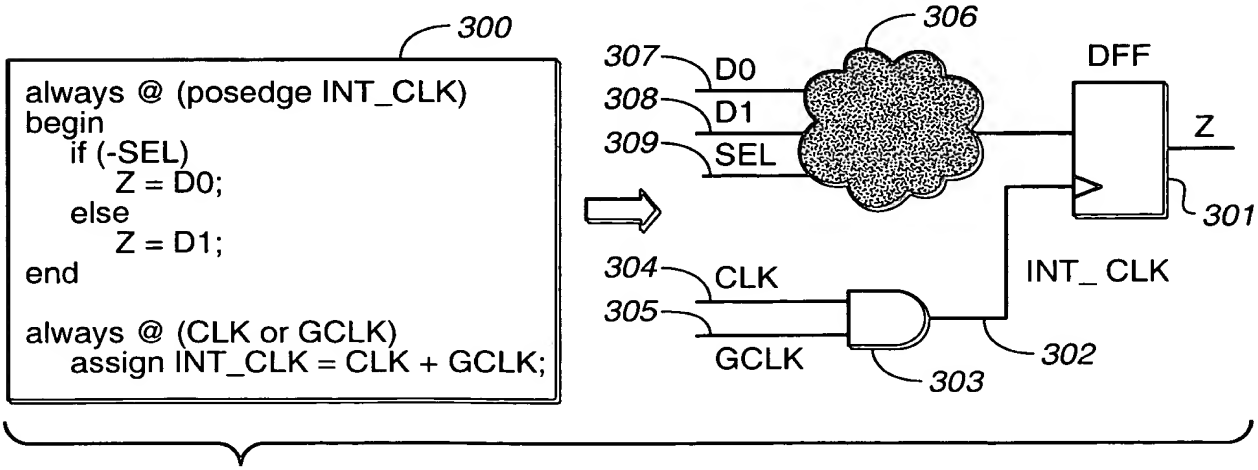


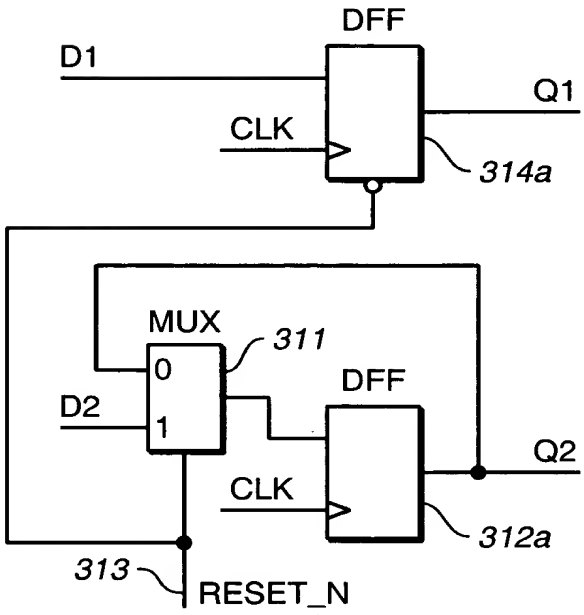
FIG._33 : Example of RTL code and equivalent. Hardware view for RTL analysis.

310

```
process (RESET_N, CLK)
begin
    if (RESET_N = '0' ) THEN
        Q1<= '0' ;
    elsif CLK' event AND (CLK = '1' ) THEN
        Q1<= D1;
        Q2 <= D2;
    endif;
and process;
```

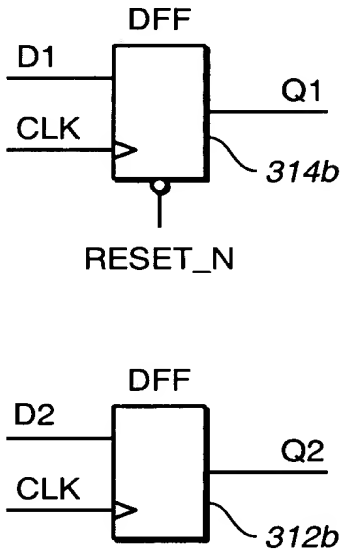
a. VHDL code for a 2-bit register with partial asynchronous reset

FIG._34A



b. Implementation created by Synopsys Design Compiler

FIG._34B



c. Implementation created by AMBIT BuildGates

FIG._34C

Implementation of partial asynchronous reset.

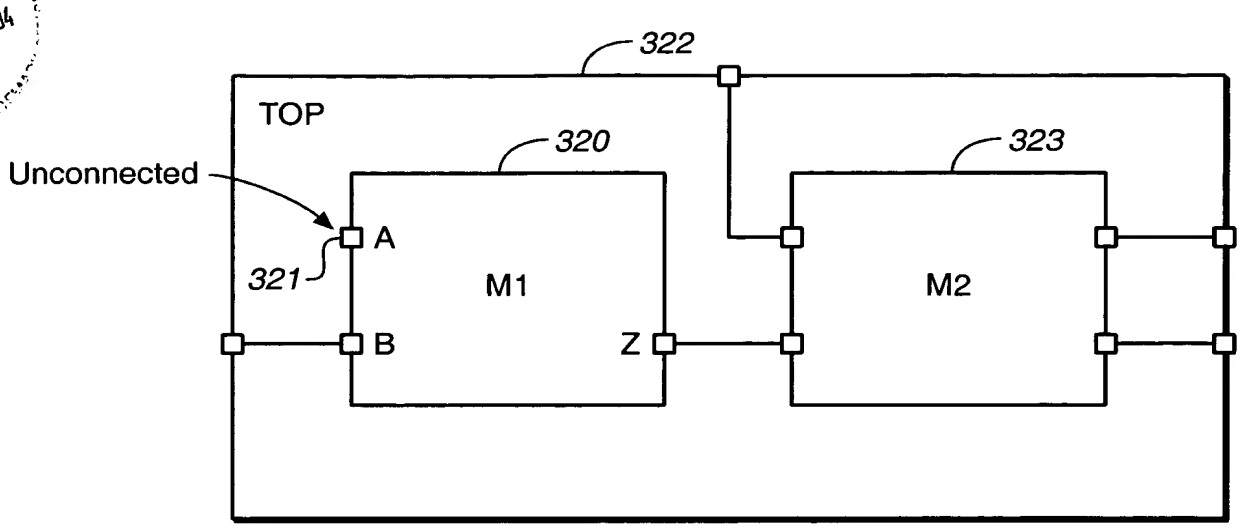


FIG._35A : RTL code

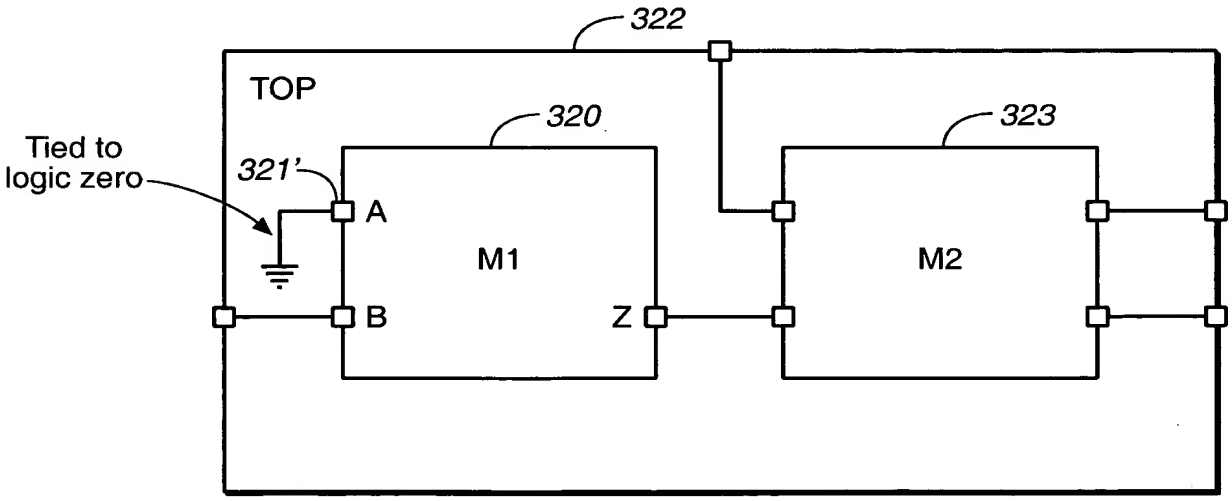


FIG._35B : Synopsys Design Compiler view of the RTL code

Handling of unconnected module input pins by Synopsys Design Compiler.

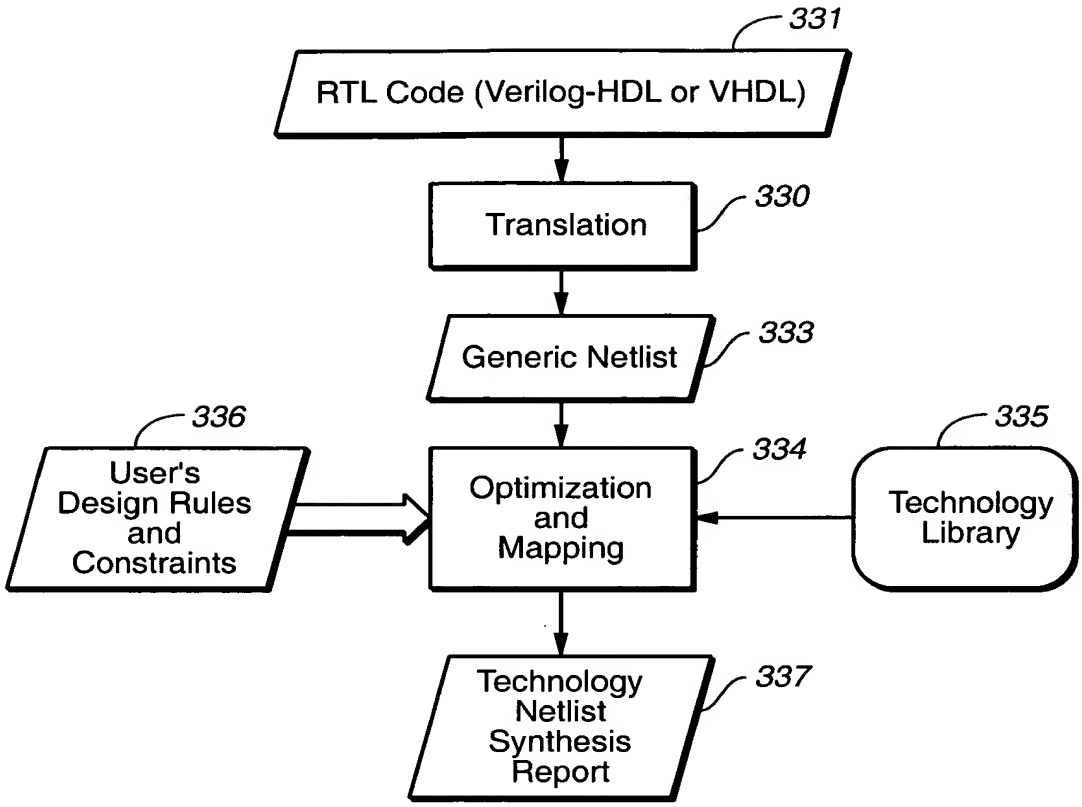


FIG._36 : Logic synthesis process

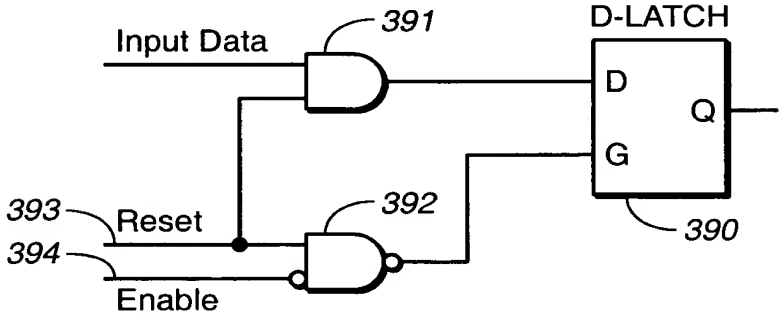


FIG._37 : Failing implementation of a latch with clear.

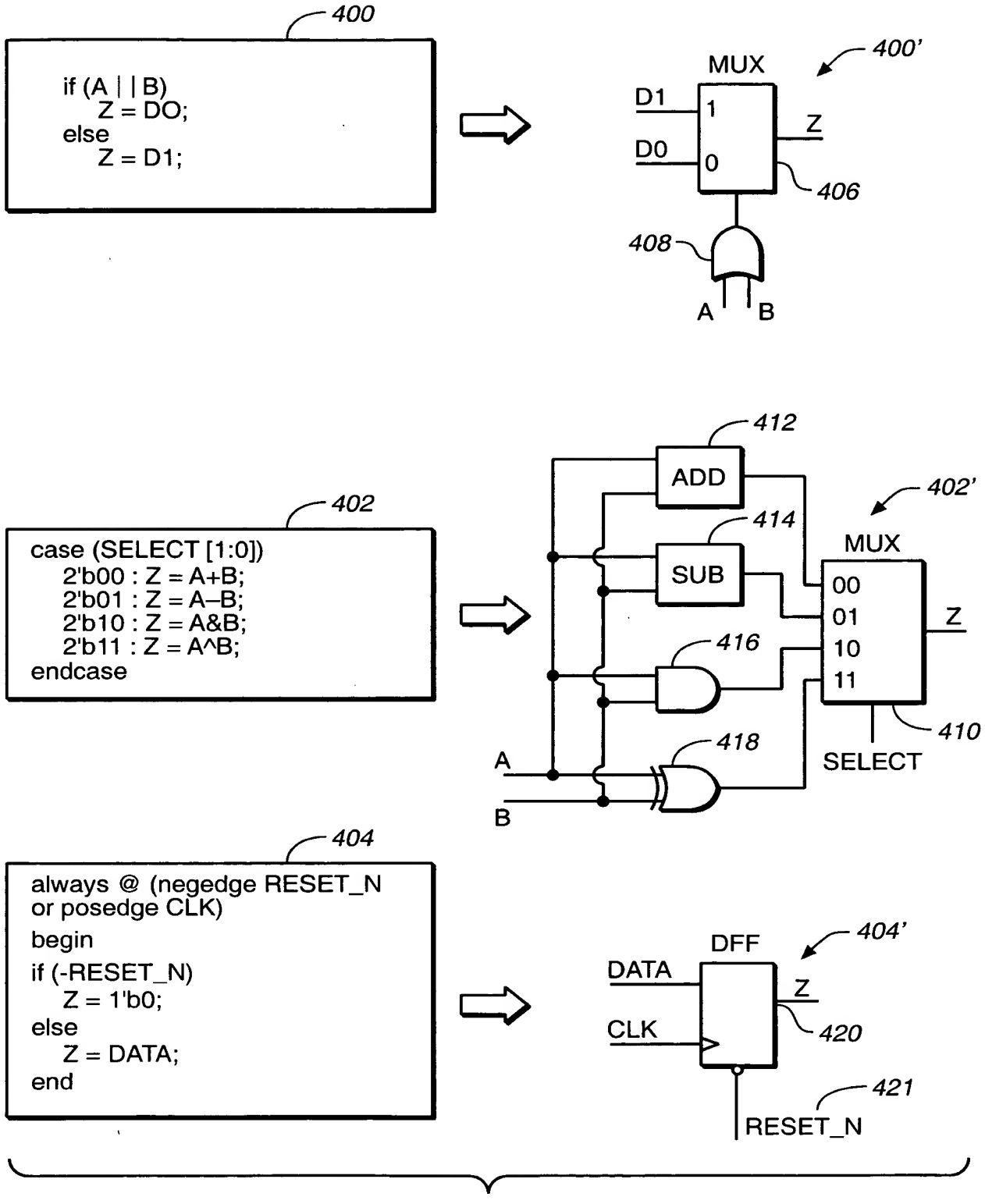


FIG._38

: Examples of transforms used for RTL code translation

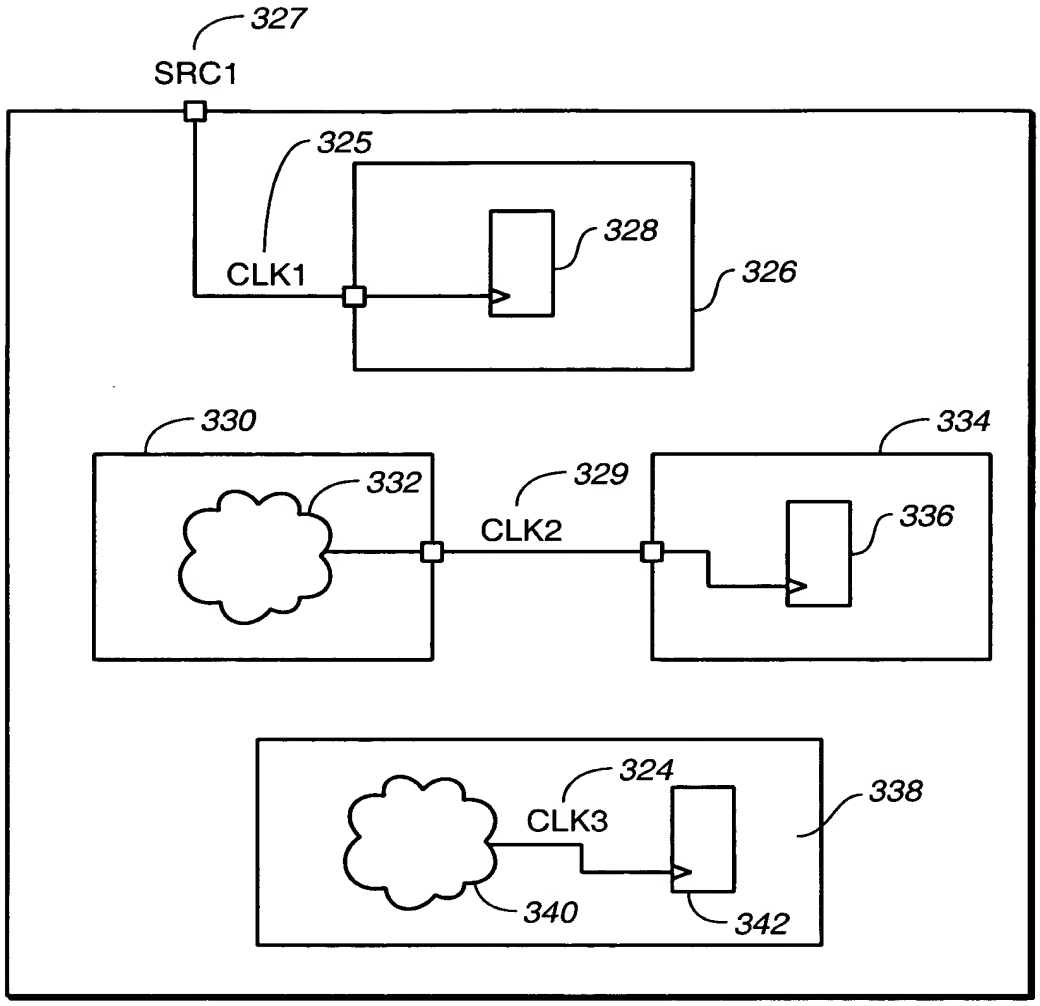


FIG._39 : External and internal clocks.

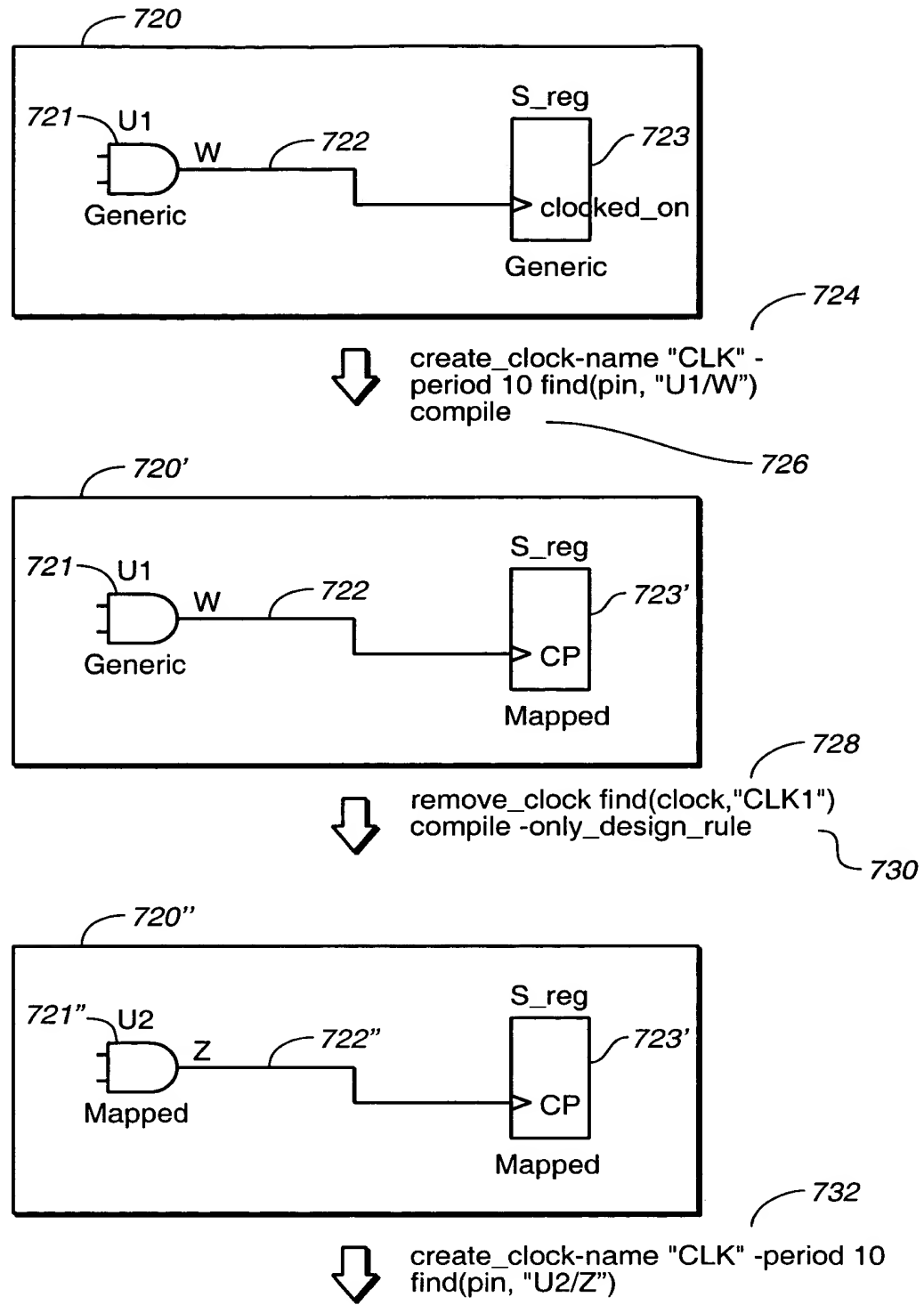


FIG._40

: Process used to map cells that create internal clocks.

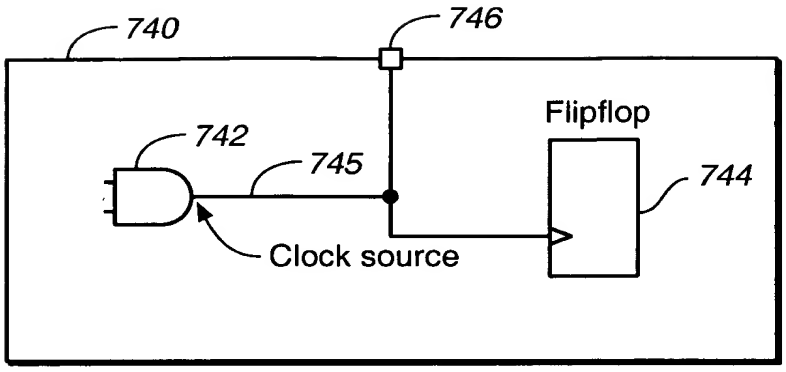


FIG._41A

: Clock retrieved through using a connected port.

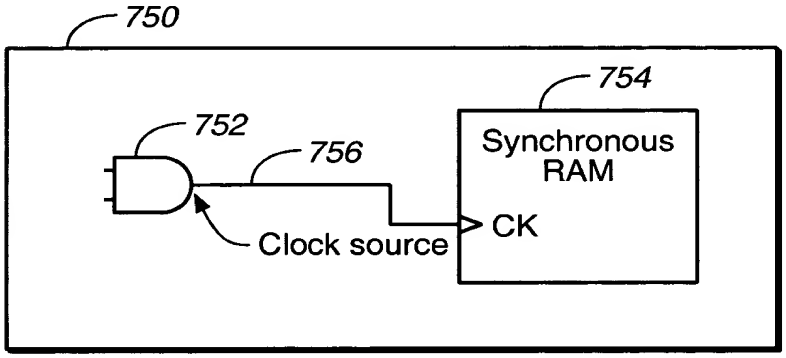


FIG._41B

: Clock retrieved through using a connected clock input pin on a RAM.

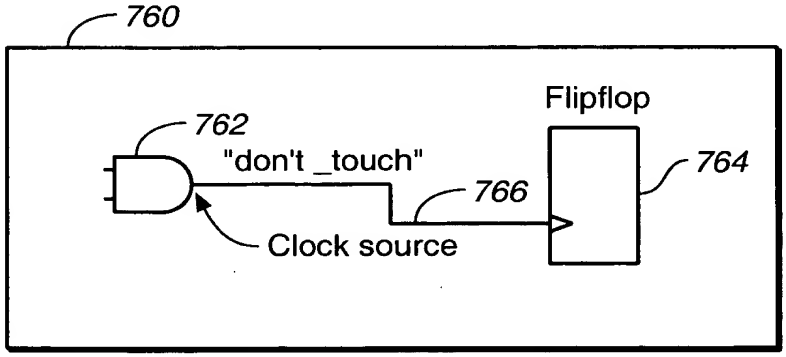
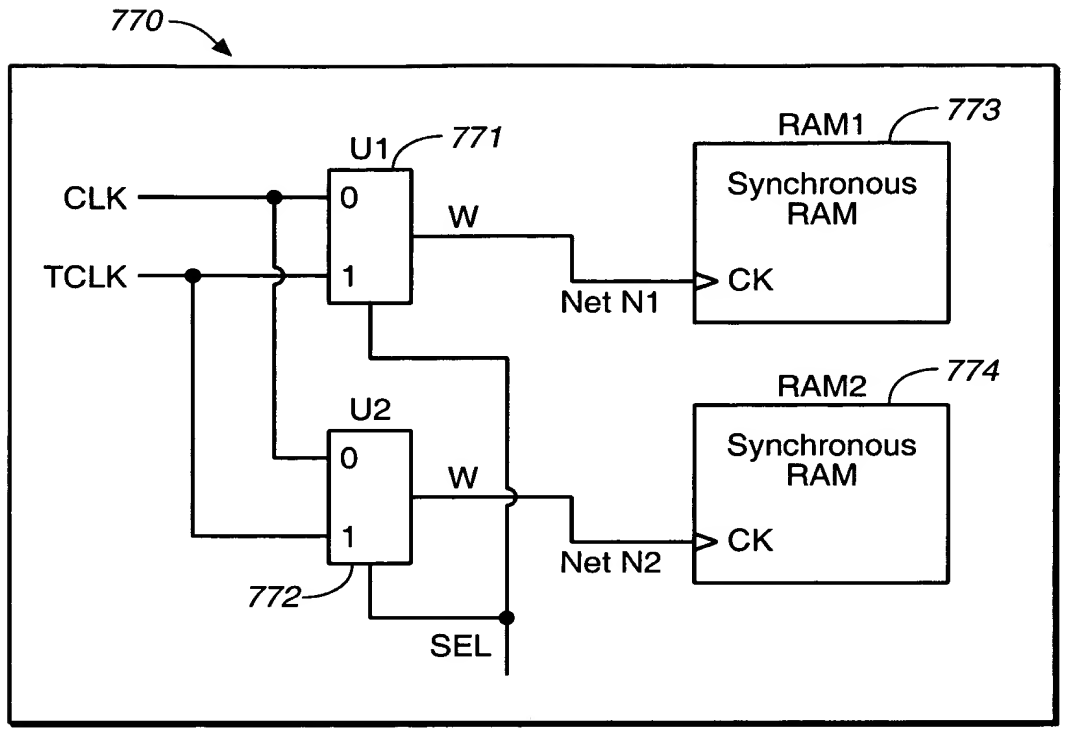
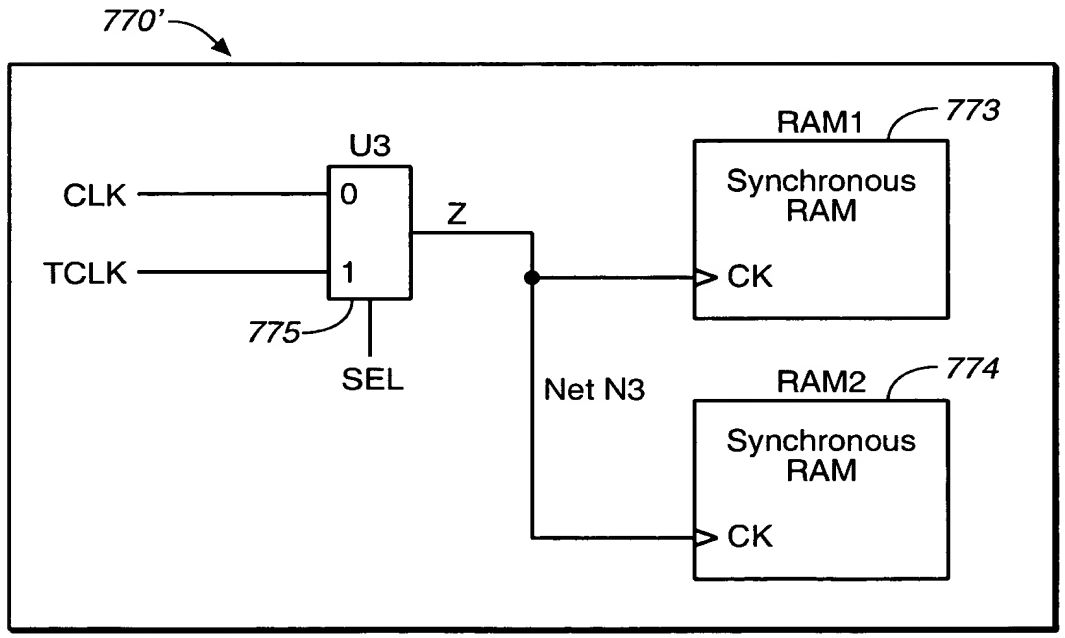


FIG._41C

: Clock retrieved through using a connected net. Retrieving names of new source pins.



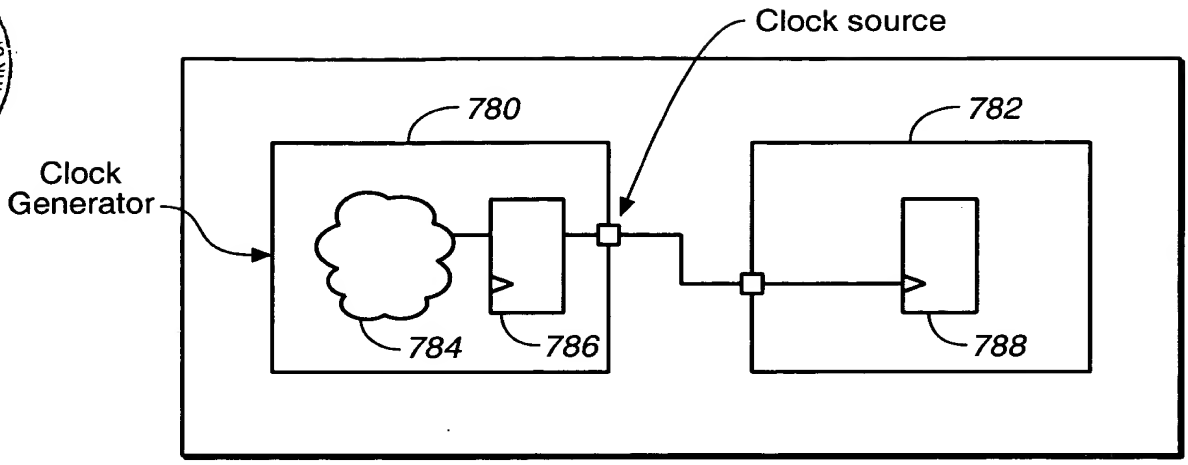
(a) Before initial mapping



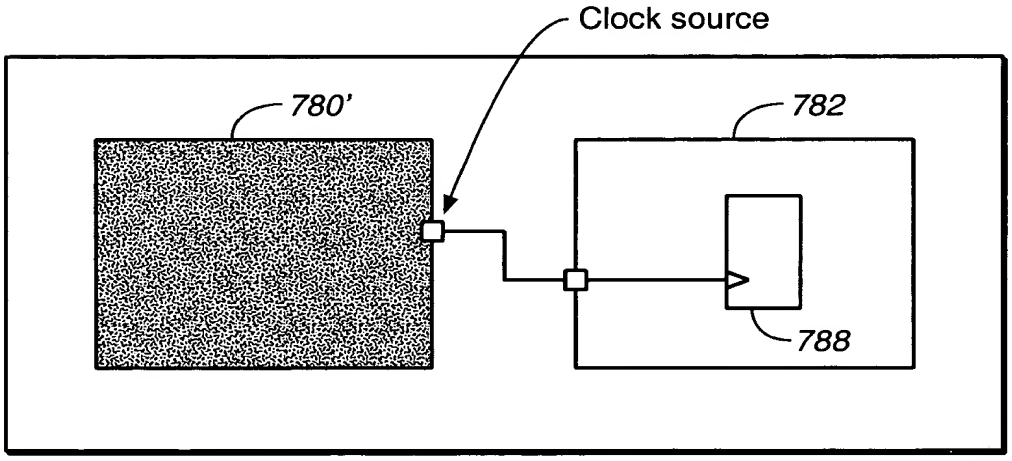
(b) After initial mapping

FIG._42

: Example of internal clocks altered through initial mapping



(a) Generic netlist



(a) After making the clock generator a blackbox for VEGA analysis

FIG._43

: Handling clock generators with a "backbox_design" directive.

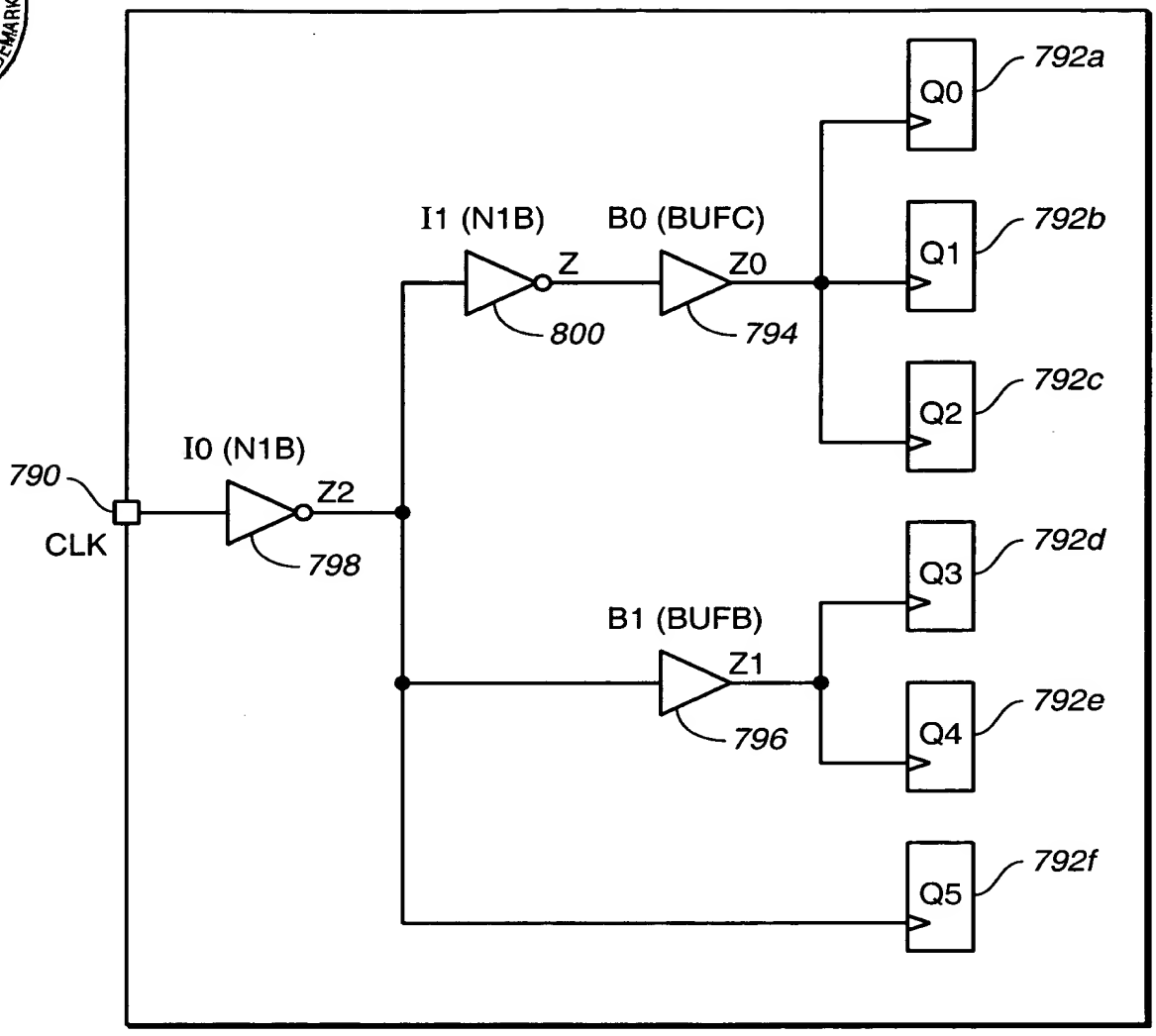


FIG._44

: Example of buffering tree used for clock distribution

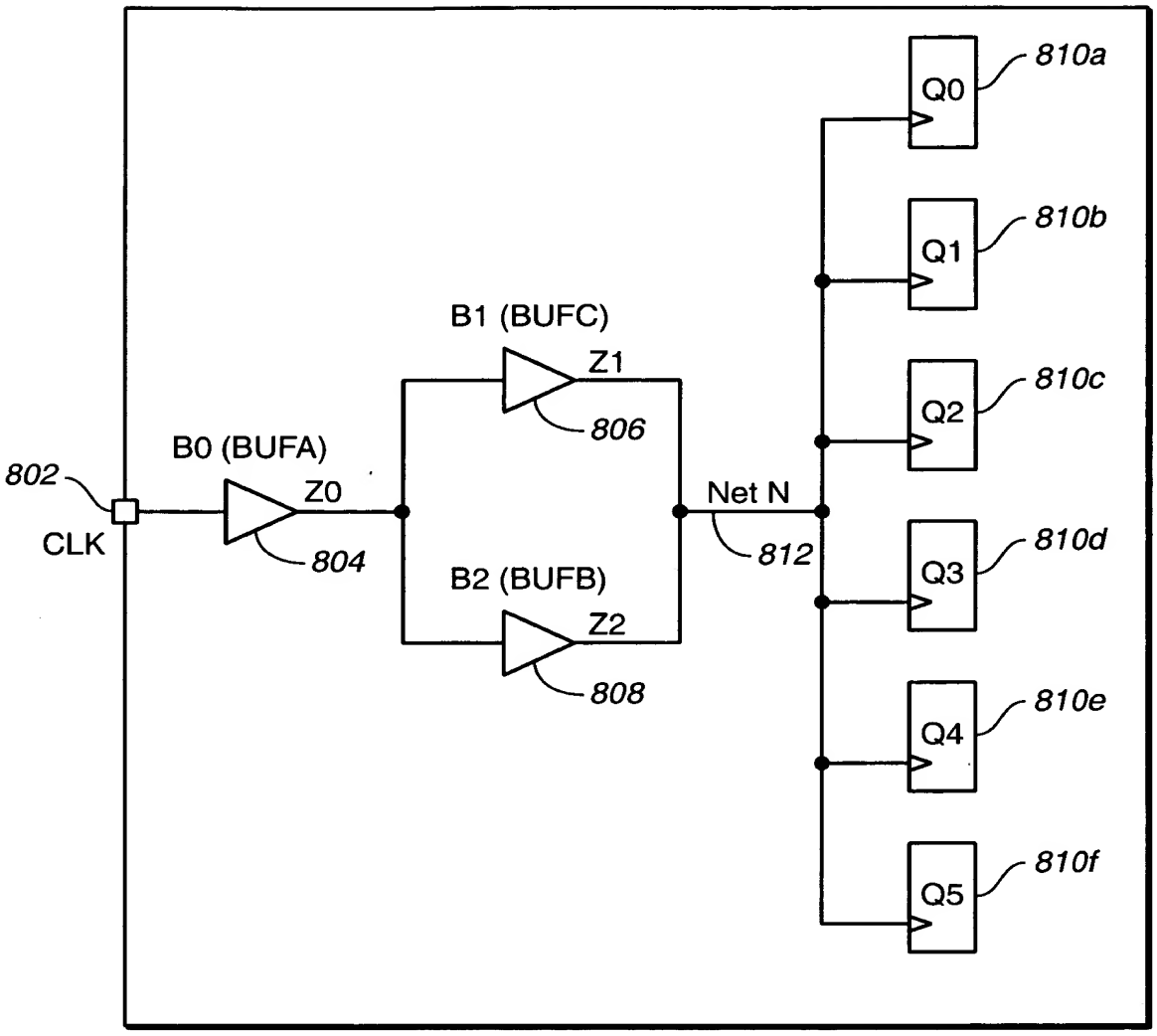


FIG._45

: Example of parallel buffers

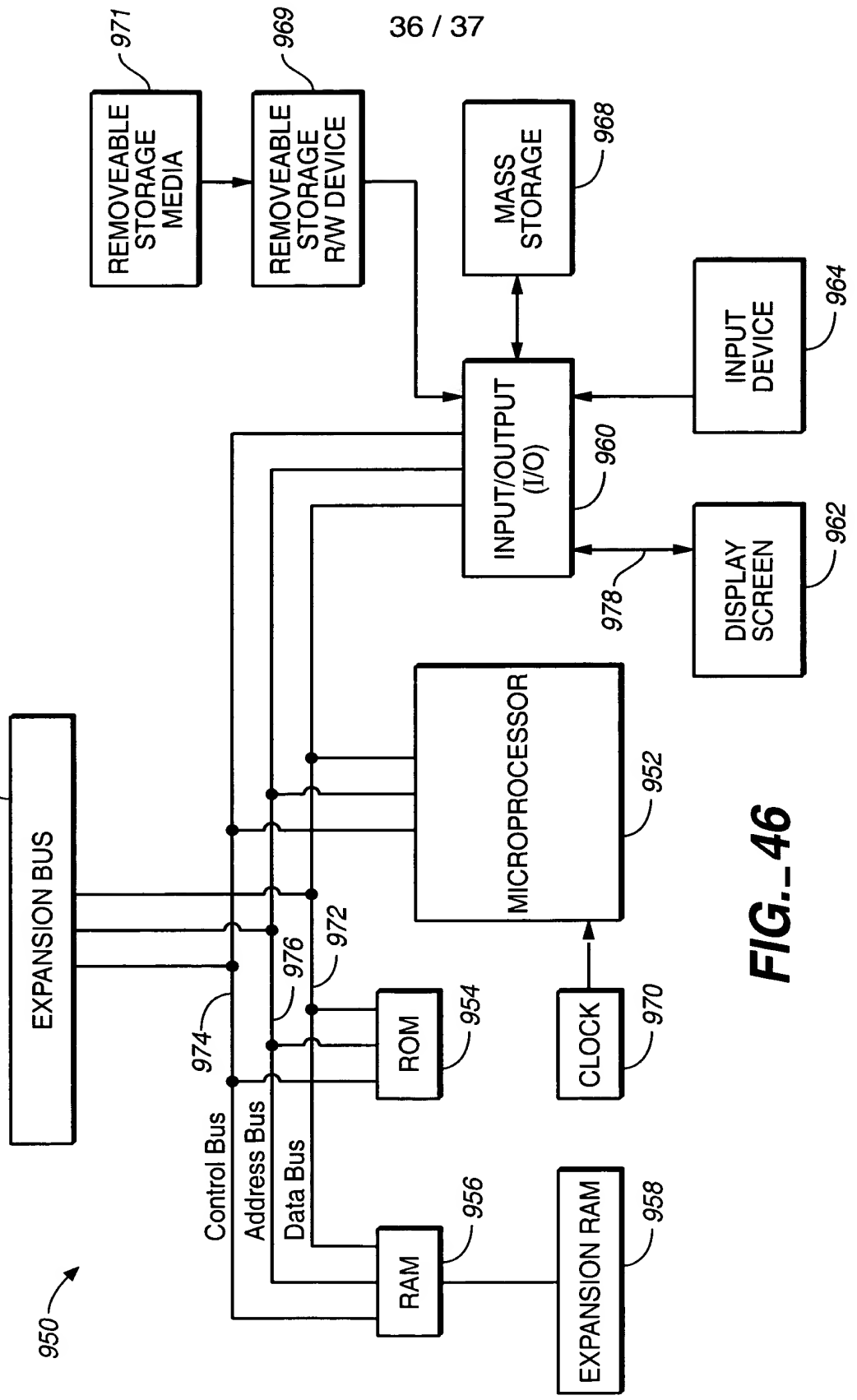


FIG._46

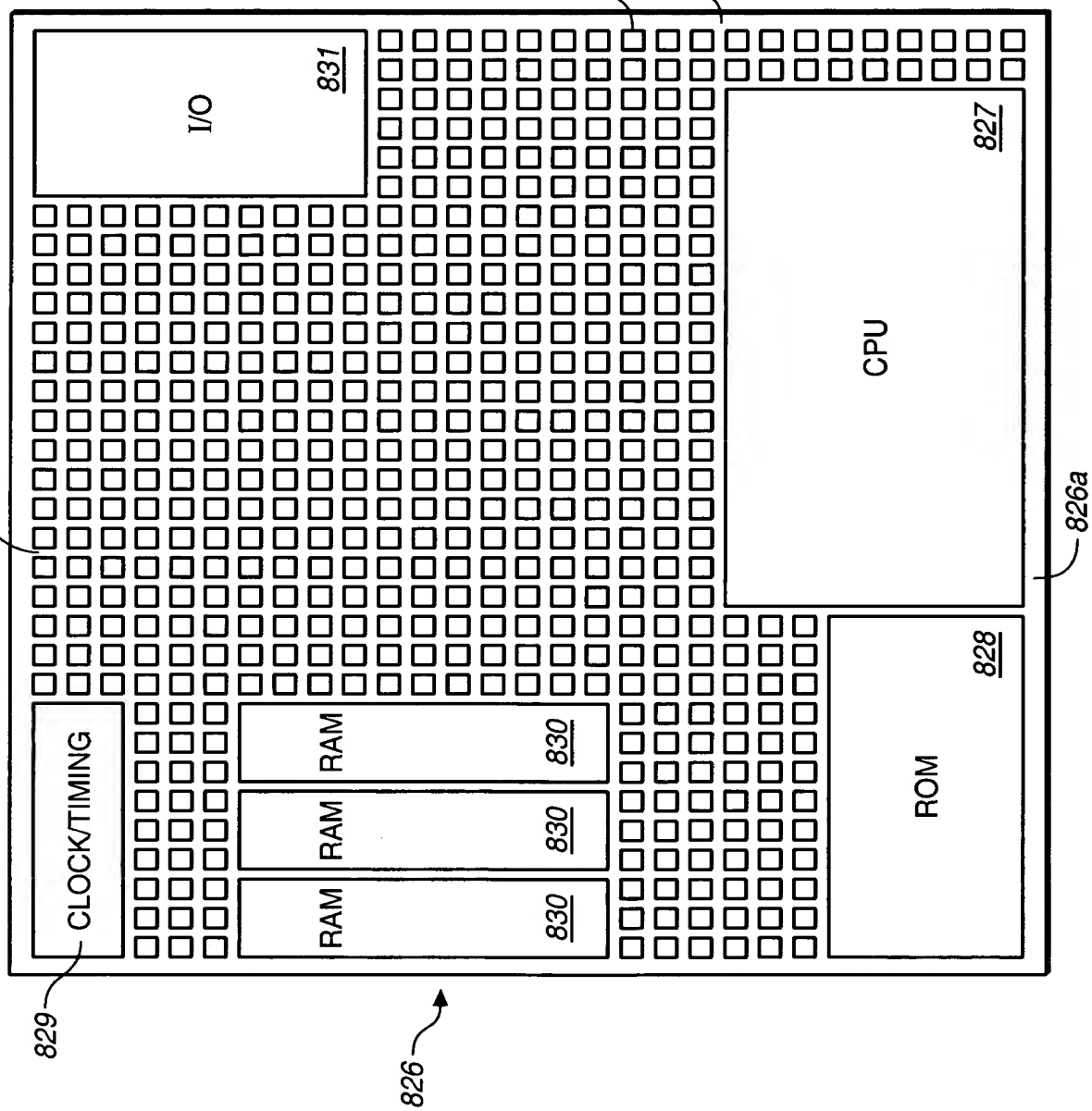


FIG._47